UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Ingeniería de Telecomunicación

PROYECTO FIN DE CARRERA

Monitorización de la calidad de señal en redes móviles basada en Android

Autor: Daniel Delgado Vico
Director: Jaime José García Reinoso

Septiembre 2011

Título: Monitorización de la calidad de señal en redes móviles basada en Android
Autor: Daniel Delgado Vico
Director: Jaime José García Reinoso

EL TRIBUNAL

Presidente:	Mario Muñoz Organero
Secretario:	Antonio de la Oliva Delgado
Vocal:	Marcelino Lázaro Teja

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 7 de septiembre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar quiero agradecer a Jaime, mi tutor, todo el apoyo prestado durante la realización del Proyecto, empezando por aceptar mi propuesta el día que aparecí por el despacho con un montón de ideas descolocadas sobre todo lo que tenía en mente. Gracias por poner todo en orden y dar sentido al Proyecto con todos los consejos que me has dado.

Ahora que escribo los agradecimientos pienso en todo el camino recorrido durante los años de Universidad y me vienen a la mente muchos compañeros que han hecho que todo el esfuerzo estuviera salpicado por momentos que me han hecho realmente feliz. Gracias por acompañarme en el viaje chicos. Especialmente tengo que sentirme afortunado de haber conocido a Iván, Patri y César.

Iván, eres un amigo incondicional y espero poder contar contigo toda la vida, creo que no he conocido mejor tipo que tú en mi vida. César, mi hermano, no tengo páginas suficientes para describir lo importante que ha sido para mí conocerte. En la carrera lo hemos pasado todo juntos, lo bueno y lo malo, y con la reconfortarte sensación de que podía contar contigo para lo que fuera, gracias. Y Patri, por todo lo que hemos compartido juntos estos años. Por no hacerme sentir un extraterrestre y compartir conmigo tantas inquietudes y la manera de ver la vida. Gracias por preguntarme aquel día dónde estaba el aula de ordenadores, porque ese día tuve la gran suerte de conocerte.

Agradecer a mis padres y a mi hermana Cristina el haberme hecho ser quien soy y por no pedir nada a cambio a todo lo que me habéis dado, sólo apoyo y amor incondicional. Os quiero.

Y por último a Laura, mi mitad. Gracias por todo el apoyo que me has dado día a día y por animarme en los peores momentos. Gracias por estar siempre a mi lado. Gracias por aguantar tantas horas de ausencia preparando este Proyecto que sin tu ayuda no podría haber realizado. Espero estar a la altura y devolverte todo el cariño que me entregas a diario. Te quiero.

Resumen

Con la llegada masiva al mercado de los denominados teléfonos inteligentes o *smartphones*, es posible disfrutar de un pequeño ordenador personal en cualquier lugar, con conexión de datos de alta velocidad y cada vez más, apto para todos los bolsillos. Esto ha permitido que se dispare la oferta de aplicaciones y servicios que demandan un alto ancho de banda y los procesadores que incluyen, permiten correr casi cualquier aplicación.

La inmensa mayoría de las aplicaciones que se ofrecen a los usuarios son unidireccionales: los contenidos los ofrece la red hacia el usuario con pequeña interacción por parte de éste, a excepción de servicios más horizontales, como los relacionados con redes sociales o voz IP, por poner algún ejemplo. Pero existe un tipo de información que puede proporcionar el conjunto de usuarios de una red al propio proveedor de servicios en **sentido ascendente**. Por separado puede no ser relevante, pero el agregado aporta una base de datos muy potente para el operador de red, siendo esta información prácticamente transparente al usuario. Estamos hablando de parámetros relacionados directamente con los estándares de telefonía como el nivel de señal, relación señal a ruido, calidad de la voz en la llamada o retardo en la recepción de la señal. Nos interesa capturar cierta información del enlace radio para que la capa de aplicación pueda manipularla y actuar en consecuencia.

Un terminal móvil mide constantemente los niveles de señal de las frecuencias que le indica la red y en este proyecto vamos a presentar la manera en que esta información pueda quedar anclada a una posición geográfica concreta gracias a la utilización de los módulos GPS que incluyen la mayoría de dispositivos del mercado, consiguiendo así no sólo saber qué terminal y en qué momento se realiza una medición de señal, sino **dónde** ha sucedido para poder posicionar con exactitud la cobertura en tiempo real de una determinada zona. Además, trataremos de detectar eventos de **llamadas caídas** (con desconexión anormal) para poder localizar, desde el punto de vista del operador, zonas con alto riesgo de interrupción de llamada, algo que repercute directamente en el usuario.

La manera en que los terminales capturan los datos de señal y eventos no queda establecida de manera estática, sino que desde la red pueden configurarse determinados parámetros en la medición: el servidor remoto será capaz de indicar a los terminales qué zona geográfica debe medirse y podrá activar o desactivar dicha medición a conveniencia.

La plataforma elegida para el desarrollo del sistema ha sido Android de Google.

Palabras Clave

Calidad, Optimización, Nivel de señal, Geolocalización, Android.

Tabla de Contenido

Índice de Figuras

1. Introducción

En este primer bloque se presentan las motivaciones que han desembocado en la realización del Proyecto y que pretenden exponer la necesidad de un sistema como el que se plantea para cubrir ciertos vacíos en la monitorización de la calidad de red. Es éste el momento de definir los objetivos que han de cumplirse para la obtención de los resultados que nos marcamos como reto al comienzo del estudio.

1.1. Motivación

Durante los años 90 y gran parte de lo que va de siglo, el objetivo de lo operadores de Telecomunicaciones y de todas las empresas relacionadas con el sector, ha sido desplegar de forma masiva la redes de telefonía móvil. Esto supuso una cantidad enorme de inversión en infraestructuras, equipos y obra civil, por no hablar de las licencias administrativas. La cuestión no era cómo, sino cuánto. Había que desplegar de cualquier forma para llegar al mayor número de clientes y poder captar la mayor cuota de mercado posible.

En un país como España esa fase de despliegue está más o menos superada (exceptuando a Yoigo) y simplemente se integran puntos en localizaciones muy estudiadas (rural con cierto número de habitantes, empresas, carreteras, AVE, etc.) y en definitiva, en todos aquellos puntos en los que la inversión tenga una **rentabilidad** asegurada. Además, poco a poco, se consolidan las comparticiones, que son localizaciones en las que despliega un operador la estación física y el tráfico se separa entre los participantes de manera lógica, es lo que se llama RAN Sharing.

Actualmente, los operadores más importantes del país ofrecen cobertura poblacional de más del 99% y la mayor parte de la inversión no se centra en desplegar puntos nuevos sino en renovar los existentes (adecuación de la red a la llegada de LTE)

Por todo esto, en los últimos años y sin duda en los próximos, la ventaja competitiva entre los operadores está en la calidad de la red y en los servicios de valor añadido que se ofrecen. Respecto a esto último, los operadores han evolucionado mucho a la hora de ofrecer gran cantidad de servicios para evitar que ese trozo del pastel no acabe en terceros, como ha ocurrido años atrás.

Sin duda, el futuro es la orientación a los servicios y estos deben ofrecerse con la mayor **calidad** posible. En prácticamente todo el territorio nacional podremos contratar con garantías los servicios de cualquier operador pero lo que realmente va a diferenciar a unos y otros es la velocidad que ofrecen, cuanto tiempo puedo estar llamando sin que se caiga la llamada, si la red se congestiona mucho o poco en Nochevieja, etc.

La importancia de conseguir estos objetivos viene motivada tanto por la demanda que generan los usuarios cómo por las garantías que exige La Administración en la explotación del servicio. En nuestro país, el Ministerio de Industria, exige unos informes trimestrales a todos los operadores (de fijo y móvil) para hacer pública la calidad de cada uno de ellos incluso estableciendo unos umbrales por debajo de los cuales las empresas pueden ser penalizadas y obligadas a compensar a los clientes. Como ejemplo de estas publicaciones a continuación se expone la comparativa de la tasa de llamadas interrumpidas del cuarto trimestre del 2010:

OPERADOR	MEDIDAS (%)
euskaltel	0,69 %
movistar	0,68 %
orange	0,71 %
vodafone	0,69 %
yoigo	0,77 %
MEDIA PONDERADA(1)	0,69 %

Figura 1: comparativa cuarto trimestre 2010. Llamadas interrumpidas [(1)]

1.2. Objetivos

En la primera parte se presentarán las diferentes maneras en que podemos monitorizar y optimizar la calidad en una red móvil y en concreto de la parte radio. Presentaremos las herramientas que poseen las empresas dedicadas a mejorar la calidad de red y propondremos una solución para tratar de abordar algunos de los aspectos menos explotados históricamente.

Como veremos en la parte del estado del arte, existen equipos sofisticados que registran todos los mensajes de la interfaz aire para caracterizar una comunicación concreta en un dispositivo móvil. La idea es tratar de simplificar el software de medida a la mínima expresión para que pueda ser ejecutado en gran parte de los dispositivos que utiliza cualquier usuario de telefonía y que reporten esa valiosa información de modo transparente al usuario.

[1] Los informes trimestrales completos pueden consultarse en la Web del Ministerio de Industria: http://www.mityc.es/telecomunicaciones/

Es el momento histórico apropiado gracias a la democratización de los terminales avanzados o *smartphones* que permiten ejecutar aplicaciones más o menos complejas aprovechando la inclusión de programas de terceros (*third party*) en las principales plataformas. Tanto es así que por primera vez, existe un sistema operativo libre como Android, que permite y facilita el desarrollo de aplicaciones que aporten valor añadido a la explotación de red y mejoren de manera sustancial la experiencia del cliente.

Es esta experiencia del cliente lo que motiva el desarrollo de unidades de medida independientes, ya que no existe mejor zona a mejorar que la reportada directamente por el cliente ya que su optimización será inmediatamente detectada por estos mismos usuarios. De esta manera se puede trazar un mapa de puntos geolocalizados que tras ser analizados indican al optimizador qué zonas deben ser prioritarias para su estudio.

El objetivo es desarrollar una aplicación que pueda ejecutarse en segundo plano en la mayor cantidad de terminales posible y que se encargue de monitorizar, de modo transparente al usuario, los niveles de señal en determinadas localizaciones aprovechando la geolocalización que facilitan los nuevos teléfonos inteligentes.

Como ya hemos comentado, los *smartphones* son cada vez más asequibles entre otras cosas, por la competencia de los operadores en el creciente mercado de datos (no sólo de voz) que sufragan en gran parte los costes de estos dispositivos. Además, los fabricantes también están librando una batalla interesante en el campo de aplicaciones y servicios y facilitan el desarrollo de contenidos para varias plataformas sin que domine una claramente (Symbian, Android y Apple iOS principalmente).

En cualquiera de los Sistemas Operativos (SSOO) elegidos para el desarrollo de la aplicación de monitorización (o en todos) tenemos que tener en cuenta que almacenar de forma instantánea los niveles de señal en el terminal supondría un consumo muy elevado en recursos del mismo, por eso se propone la ejecución de la aplicación en “modo standby” a la espera de que se disparen determinados eventos que son los que realmente interesan en la optimización de la calidad. Es decir, es suficiente con esperar un evento de llamada caída, error de conexión o descarga de datos, para almacenar la información concreta asociándola al punto geográfico determinado mediante posicionamiento GPS, A-GPS o por triangulación de celdas, elegido por el cliente y previa autorización[2]. El último paso sería el envío de esta información a un servidor central por medio de un SMS o conexión de datos transparente al usuario.

En un segundo modo de trabajo, el terminal se encontrará en modo de espera hasta que se encuentre en una zona geográfica concreta, ordenada por el operador. En ese momento se almacenarán todos los puntos por los que pase el móvil, anotando la posición y el nivel de señal, entre otros parámetros radio.

[2] En principio la monitorización está pensada para grupos corporativos o académicos donde la autorización es implícita. Por ejemplo, en terminales de los propios empleados del operador o empresa de autobuses de una ciudad. Aunque también en particulares si se acuerdan los términos de privacidad.

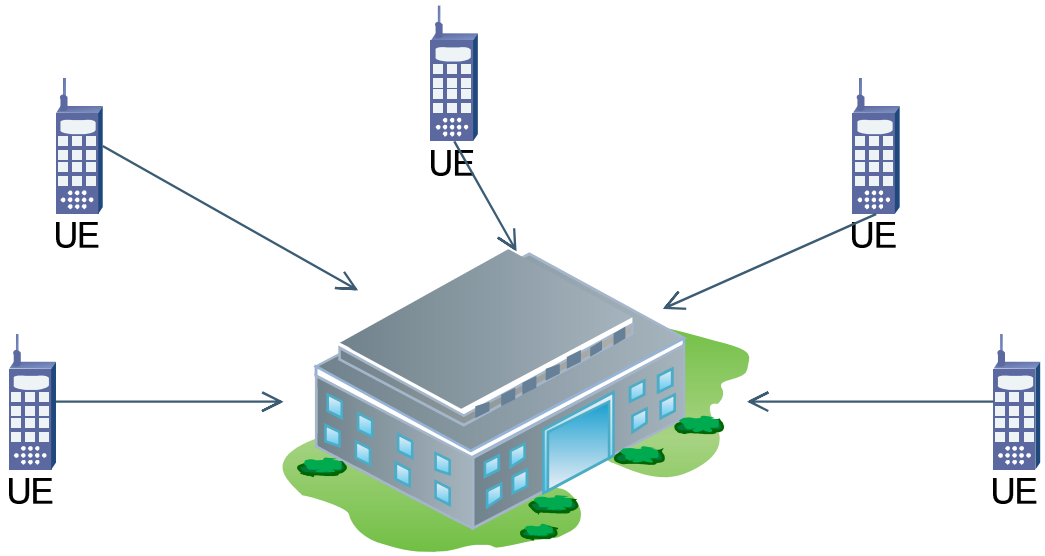

Figura 2: Solución cliente-servidor

De esta manera podremos construir un mapa en un área geográfica concreta en la que localizar puntos sensibles de señal y susceptibles de mejora, ya sea por optimización, por despliegue o por detección de incidencias de red. Por ejemplo, mapeamos las llamadas caídas en los terminales dados de alta en el servicio en la Comunidad de Madrid (figura 3)

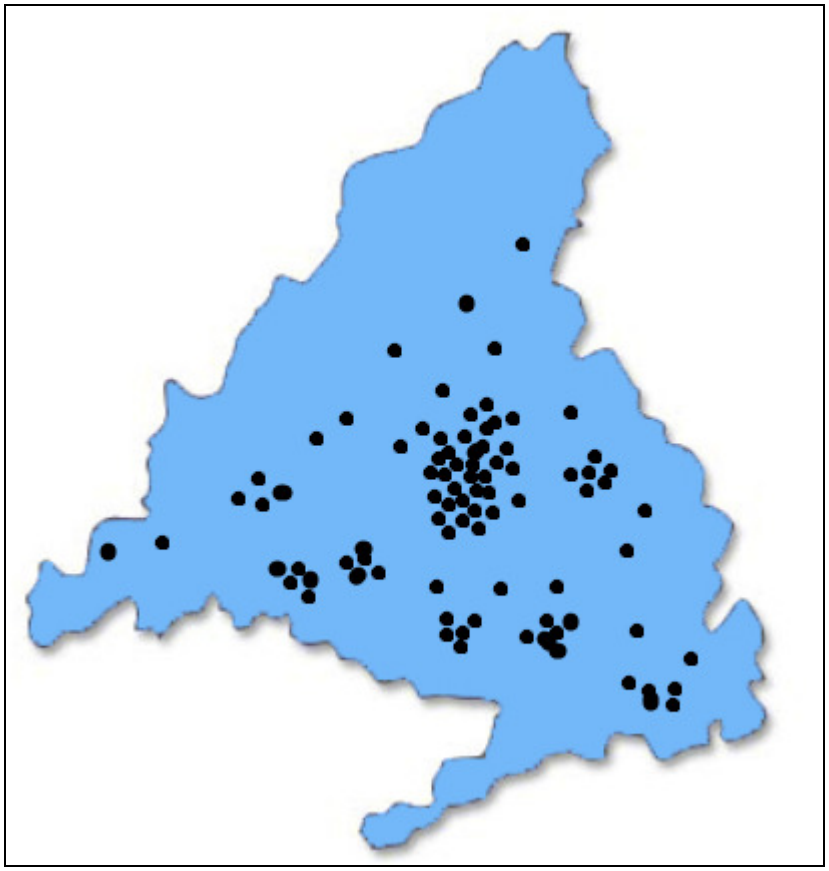

Figura 3: Mapa de llamadas caídas reportadas por los terminales

De un solo vistazo se pueden encontrar zonas de riesgo de llamada caída o de bajo nivel de cobertura.

Por todo esto, en el Proyecto fin de Carrera nos marcamos como principal **objetivo:**

Ofrecer al operador de red herramientas para la **monitorización de la calidad de red** de forma **distribuida** y **geolocalizada**, basando el reporte en **medidas de señal** y eventos de **llamada caída** ofrecidos directamente por dispositivos móviles.

Para lo cual se buscará:

- **Registrar** niveles de **señal** ofrecidos por los dispositivos móviles y asociarlos a una **posición geográfica** concreta.
- **Almacenamiento** de dicha información y **envío** a servidor central.
- Activación y desactivación, en cada uno de los terminales implicados, de medidas **bajo demanda** en zonas concretas ordenadas por el servidor.
- **Detección** de eventos de llamada caída por parte del móvil y comunicación inmediata al servidor central.

2. Estado del Arte

En este apartado se presentan los 4 principales métodos de monitorización de la calidad de red empleados por los operadores. Cada uno aporta información diferente al ingeniero radio, aunque como veremos, todos presentan algunas ventajas e inconvenientes. Una vez repasados estos métodos podremos comprender de forma clara por qué el Proyecto que se plantea cubre algunas necesidades nunca explotadas históricamente.

2.1. Estadísticas de red

Determinados elementos de red son capaces de almacenar un gran número de contadores específicos que se incrementan una vez se detectan determinados eventos. Es la manera en que podemos obtener, en poco tiempo, una gran cantidad de información de la "*performance*" de red y de su histórico.

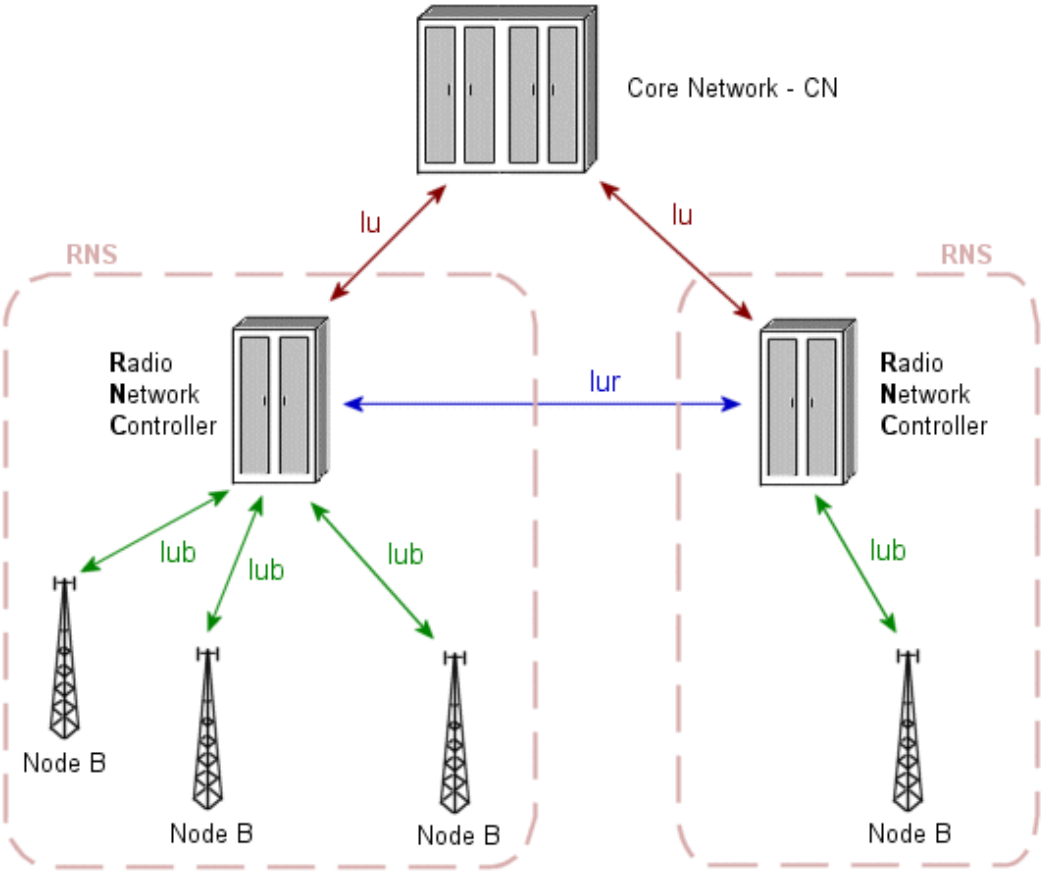

Figura 4: Arquitectura de Red UMTS

Los proveedores de equipos y servicios de red, ofrecen a los operadores una serie de funcionalidades entre las que se encuentran todas aquellas dedicadas a la

monitorización de red basada en contadores. A partir de estos contadores pueden construirse indicadores de calidad a partir de fórmulas más o menos complejas y que aportan al cliente una visión más intuitiva y amigable del comportamiento de la red. A estos indicadores los llamamos KPI, *Key Performance Indicators*. En este contexto, entendemos como cliente al operador. En la siguiente figura se muestra un ejemplo de creación de contadores de "paging" (azul):

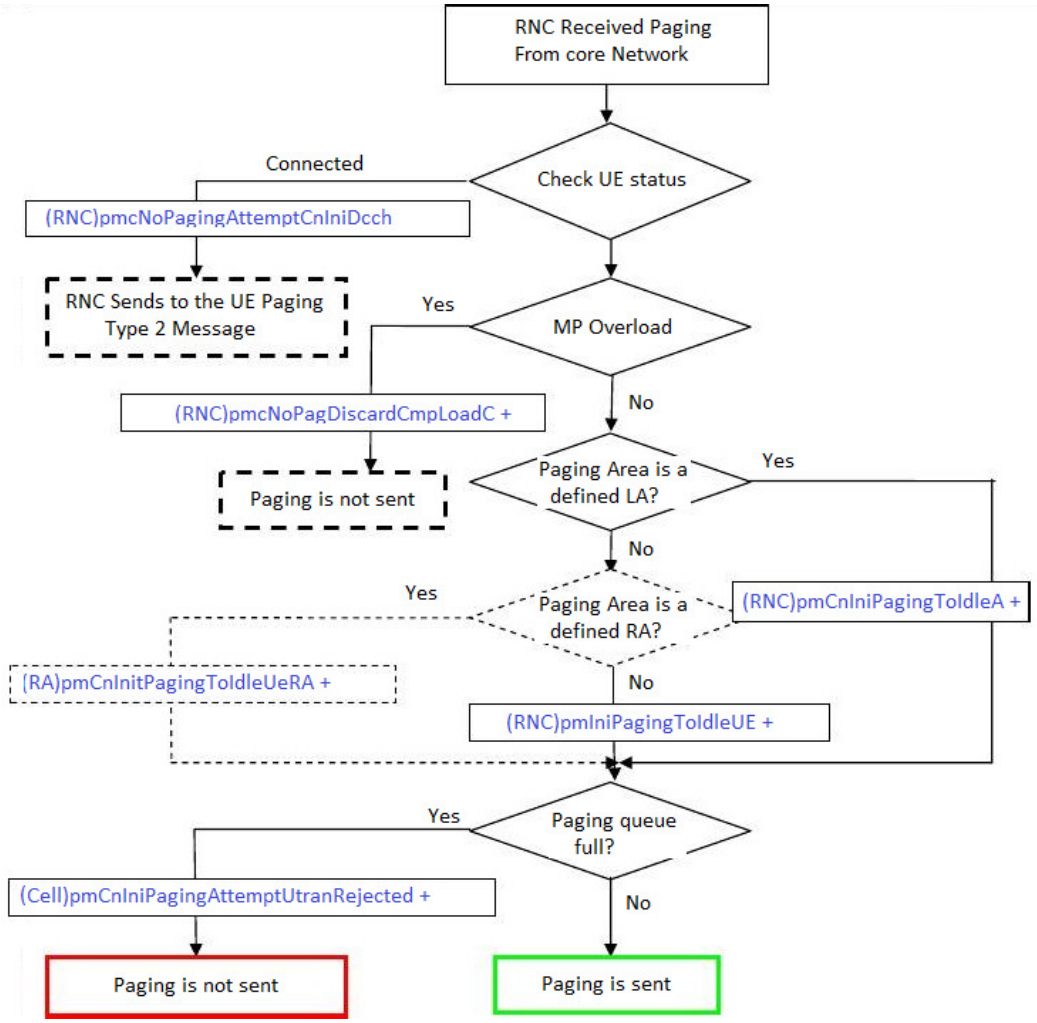

Figura 5: Ejemplo real de Diagrama de generación de contadores de "Paging"[14]

Los contadores acumulados por los elementos de red pueden generar información de varios cientos de Gigabytes diarios por lo que no suelen estar activados todos de inicio, sino que el cliente los contrata bajo demanda. Así, un contador básico puede ser el encargado de incrementar los intentos de conexión de canal TCH pero un gestor de red puede no necesitar información acerca de congestión GPRS si no le resulta crucial en su estrategia comercial.

De esta manera, en intervalos de tiempo establecidos, el conjunto de contadores se almacena en un sistema externo creando una gran base de datos vinculada donde podemos generar fórmulas más complejas, agregándolas por jerarquía de red, tiempo, provincia, etc.

Se recogen estadísticas en todos los interfaces, tanto de la red de acceso como en Core. Para el tipo de estudio que estamos realizando, nos centraremos en las estadísticas de celda analizando un caso práctico (muy sencillo). Los eventos que disparan los contadores, aunque sean referentes a eventos radio, los gestionan los elementos de agregación, tanto BSC como RNC.

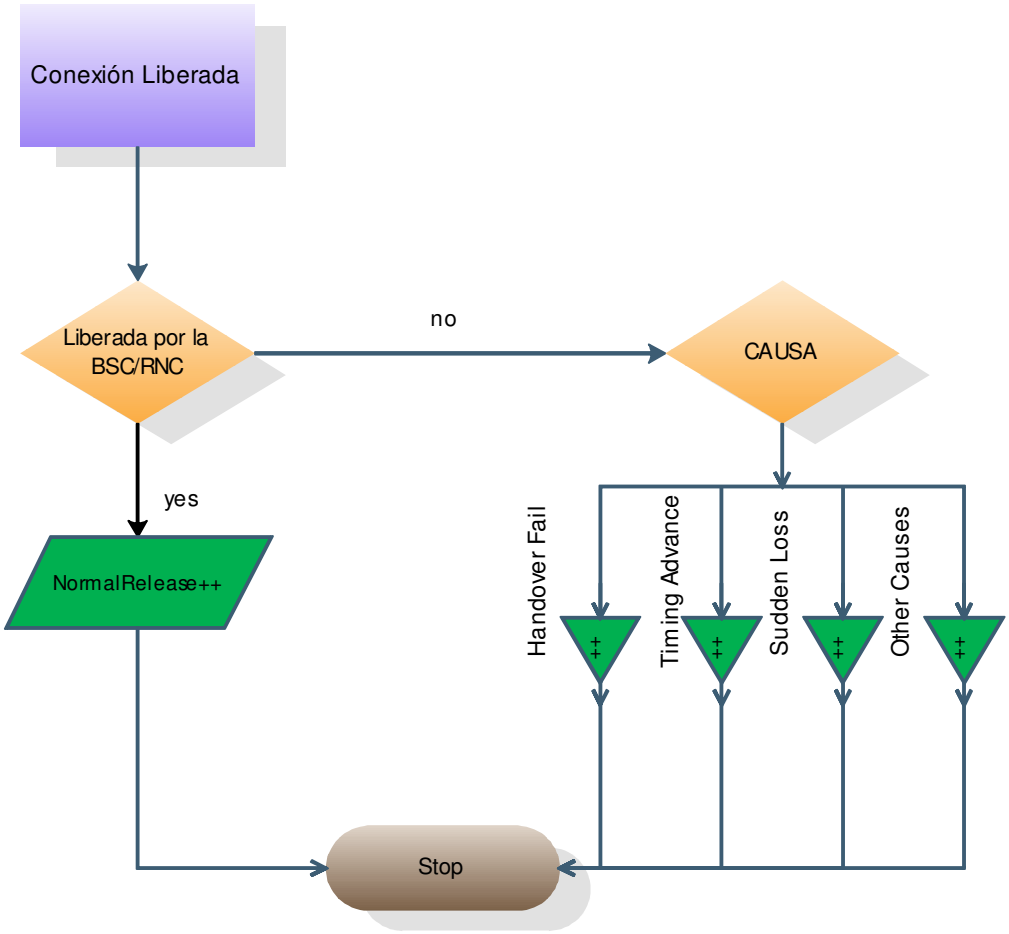

Figura 6: Ejemplo esquemático de generación de contadores de llamada caída

Veamos un caso para fijar todos estos conceptos. En la celda A, que cubre un área geográfica determinada, se incrementa el contador de "llamadas_iniciadas" cada vez que un usuario consigue un canal de voz que gestiona esa celda.

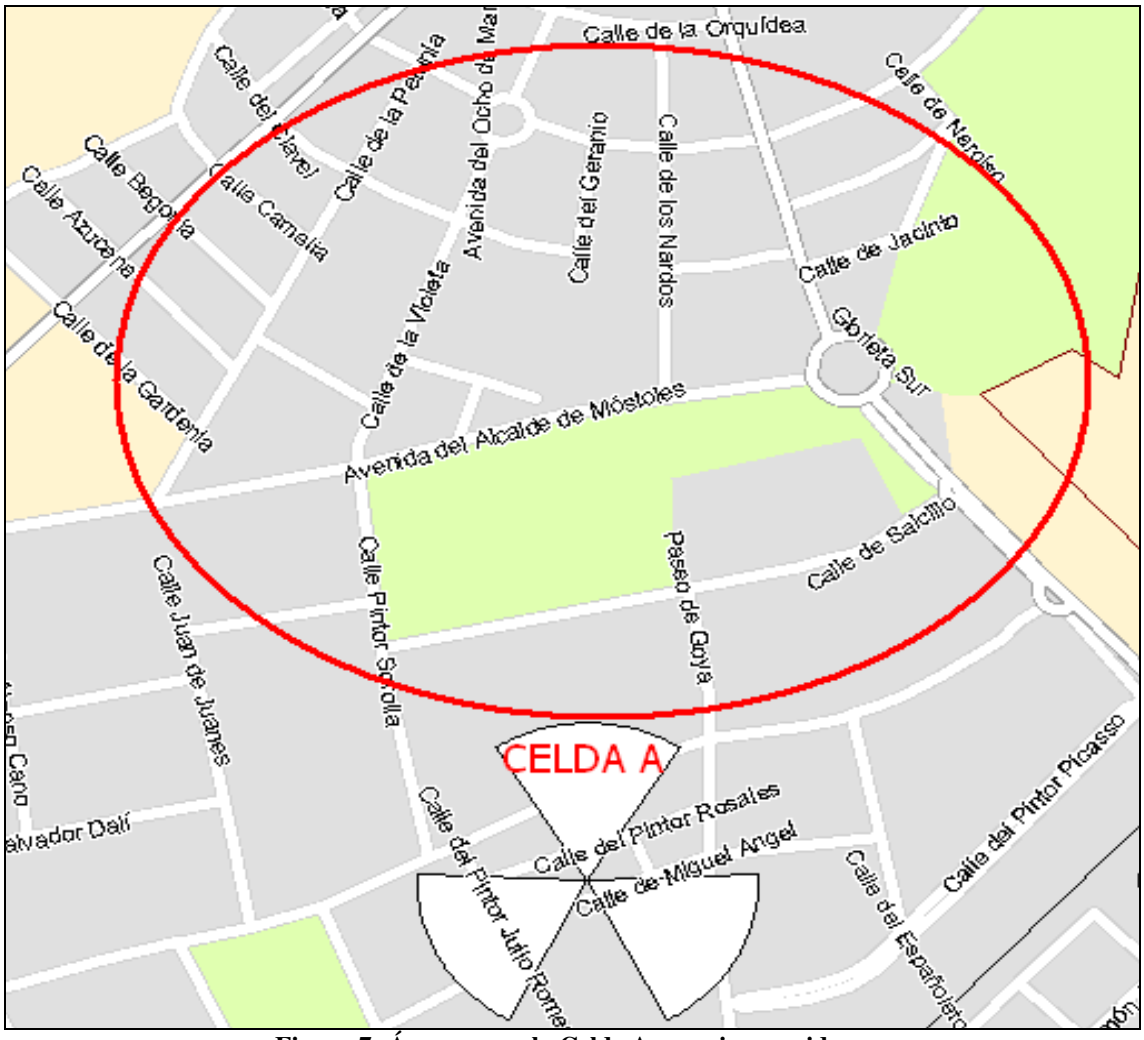

Figura 7: Área en que la Celda A es mejor servidora

A su vez, en las ocasiones en que la conexión se libera de forma inesperada, se incrementa el contador de "llamadas caídas". De esta manera, podemos disponer de dos contadores brutos a partir de los cuales construir un KPI básico: *Drop Call Rate*, que no es más que la tasa de llamadas caídas de la celda:

DCR = llamadas caídas / llamadas iniciadas

Con una aplicación capaz de generar ordenes SQL (por ejemplo) contra la BBDD matriz, realizamos consultas para obtener los contadores deseados y a partir de ellos realizar minería de datos con el objetivo de acotar y analizar el comportamiento de la estación.

En formato de gráfica, para nosotros la celda presenta este comportamiento en el intervalo de estudio:

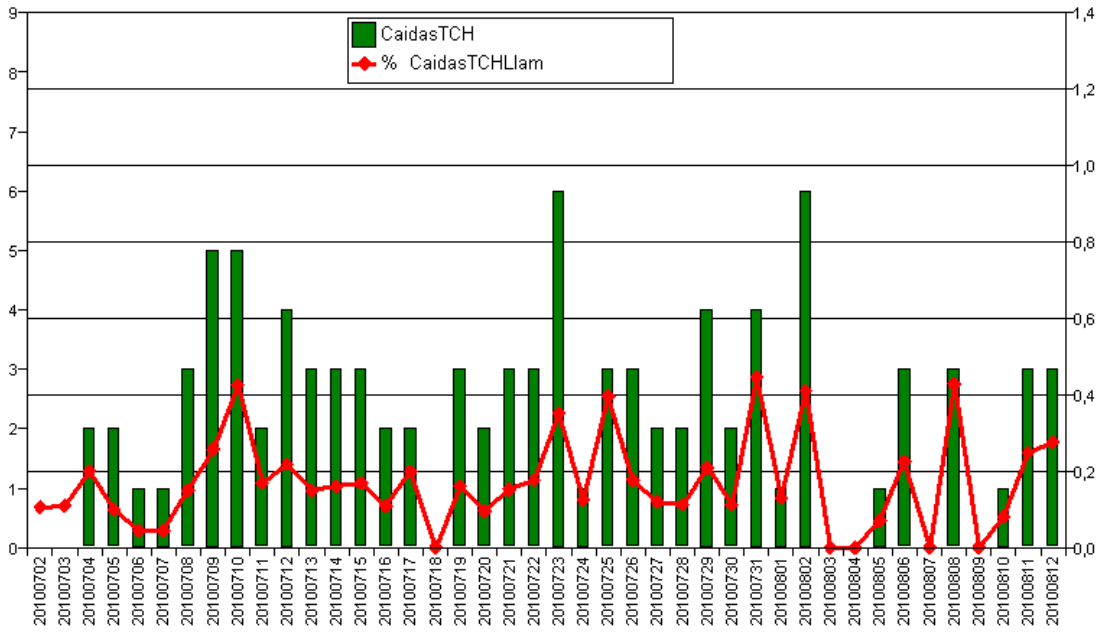

Figura 8: Evolución temporal de caídas de la Celda A

Con estas sencillas consultas, para toda la red de acceso del operador podemos trazar un mapa geográfico de la tasa de llamadas caídas por zona y centrarnos en aquellas estaciones que están causando una degradación considerable para proceder a su optimización.


Ventajas de la Monitorización de red mediante estadísticas:

- Gran cantidad de información actualizada acerca del comportamiento de la red consiguiendo una visión temporal y jerarquizada de la misma.
- Detección de incidencias inmediata. Si una estación tiene un TRX averiado y no ha saltado ninguna alarma, posiblemente su tasa de caídas se disparará, y estableciendo ciertos umbrales, puede detectarse mediante estadísticas.
- Funcionalidad que proporciona el suministrador: Llave en mano.


Desventajas:

- Necesaria una infraestructura complicada de servidores y aplicaciones capaces de almacenar y gestionar una cantidad enorme de información que se genera diariamente.
- Resolución de celda en ocasiones insuficiente para acotar problemas de cobertura o congestión muy localizados.
- En definitiva, visión desde la red, no desde el punto de vista del usuario.

2.2. Drive test

Supone una metodología completamente diferente a la del punto anterior. El objetivo es realizar medidas en recorridos concretos con los dispositivos necesarios para caracterizar el comportamiento de la red en localizaciones determinadas que con la recolección de estadísticas no somos capaces de diferenciar.

Se trata sencillamente de realizar llamadas reales con dispositivos específicos que permiten monitorizar la interfaz radio y almacenarla en *logs* que posteriormente pueden ser analizados. Estas medidas se hacen fundamentalmente utilizando vehículos preparados.

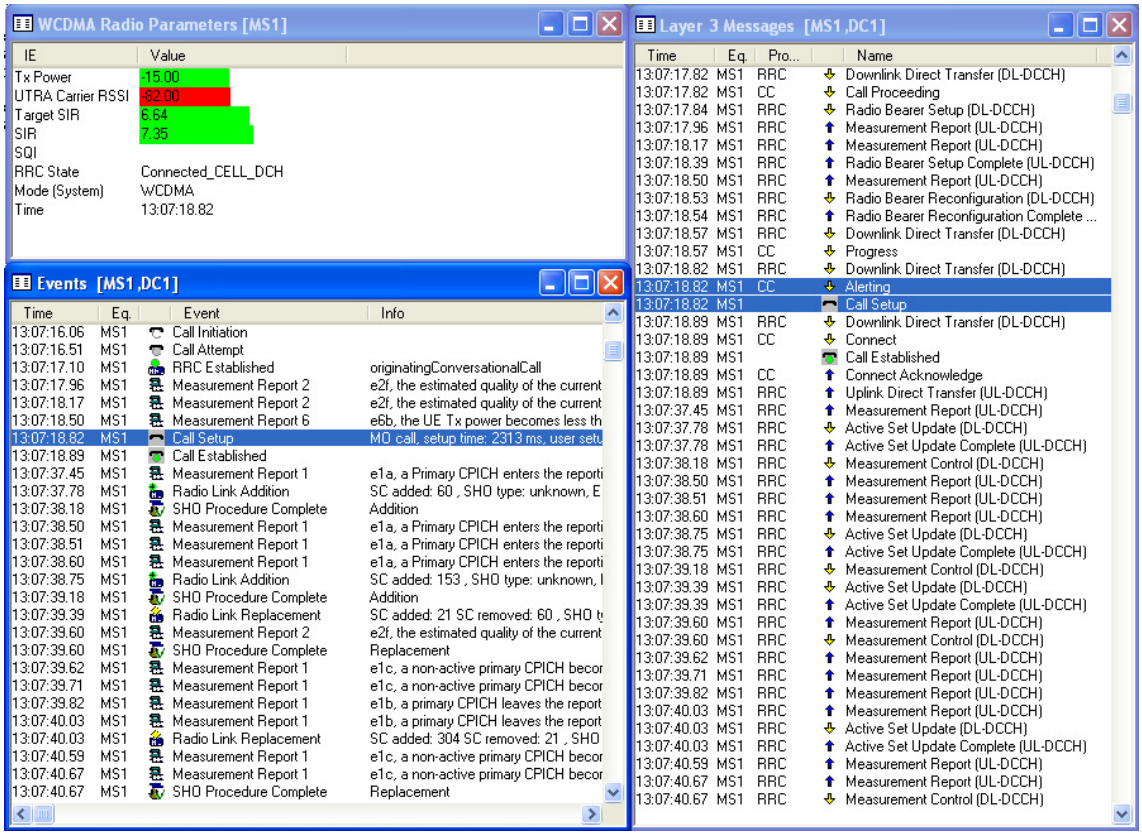

Figura 9: Captura de mensajes en una aplicación de Drive test

2.2.1. Elementos necesarios para un Drive Test

Ordenador Portátil

En él tendremos instaladas todas las aplicaciones necesarias para almacenar y presentar las medidas realizadas mediante ficheros o logs que se generan en los terminales. Dispone de herramientas de lectura de eventos, de niveles de señal, incluso de posicionamiento en mapas.

Los teléfonos van conectados al ordenador y desde el mismo se configuran las medidas, se lanzan las grabaciones y se almacenan los ficheros de salida.

Figura 10: Equipo de Drive test integrado en el vehículo

Terminal GSM/UMTS

Aparentemente son terminales generalistas que pueden encontrarse en las tiendas pero incorporan modificaciones que los permiten interactuar con herramientas específicas de drive test. Esta particularidad hace que su coste sea de varias veces el que tendría el modelo estándar en el mercado.

Lo que hace que estos teléfonos sean especiales es la cualidad de tener los puertos abiertos por lo que desde el portátil pueden monitorizarse todos los mensajes que se intercambian con la red y de esta manera una llamada de voz convencional puede estudiarse a todos los niveles mientras que para el usuario es completamente transparente.

Scanner

Dado que los terminales únicamente leen las frecuencias que les indica la red, resulta imprescindible aprovechar el drive test para leer los niveles de señal de todas las frecuencias para poder determinar si alguna vecina importante no se está monitorizando o caracterizar una celda concreta independientemente de que el móvil se enganche o no a ella. El scanner es diferente para GSM y UMTS y normalmente va acoplado al techo del vehículo. Obviamente conectado al ordenador principal.

GPS

Todas las medidas recogidas sólo podrían presentarse en histogramas o cronogramas si no fuera por la inclusión de GPS en la medida. Para cada muestra, el fichero de salida asocia, mediante una marca, una posición GPS. Esto permite presentar

los resultados en el área geográfica concreta de la medida.

Vehículo

En la mayoría de los drive test se pretende abarcar un área geográfica grande y por eso se utilizan vehículos que permitan la inclusión de una antena Scanner y la alimentación suficiente para aportar energía al resto de dispositivos.

Opcional: dispositivos *pocket.*

Para medidas en interiores no es posible utilizar un vehículo por lo que la ejecución se hace algo más aparatosa. Tradicionalmente se han tomado muestras directamente con el portátil en la mano y los terminales GSM y UMTS sujetos de cualquier forma. Al abandonar el vehículo perdemos la posibilidad de realizar medidas del scanner y la posición GPS, pero para eso pueden introducirse de manera manual *filemarks* que indican la posición relativa de algunos elementos: “Puerta de entrada”, “salón de actos”, etc.

Para mejorar esta metodología es posible utilizar teléfonos que incluyen un software simplificado del que se utiliza en el ordenador. Eso hace las medidas mucho más sencillas ya que el técnico únicamente debe llevar consigo el terminal para posteriormente volcar los ficheros de salida en el portátil. Estos dispositivos, obviamente, son mucho más caros.

2.2.2. Procedimiento de un Drive Test

En primer lugar se configura la secuencia de pasos que va a lanzar directamente el ordenador y que reducen al mínimo la interacción con los teléfonos por parte de los técnicos. Incluso si la llamada se cae, el reintento se hace de manera automática. Ejemplos:

1. Programamos llamadas continuas tanto en 2G como en 3G para un recorrido largo en la M40. Al lanzar el script, la medida sólo consiste en dirigir el vehículo por la zona requerida ya que los terminales se encargan de descolgar, llamar (a un tono piloto o teléfono de pruebas) y de recuperar la llamada en caso de perder la conexión. Al finalizar la medida tendremos una muestra real de los niveles de señal de la carretera en ambas tecnologías. *Nota: Para medidas en carretera es muy recomendable medir ambos sentidos.*

2. Detección de fallos de conexión: En lugar de movernos, programamos 100 llamadas de 15 segundos que se lanzarán de forma automática almacenando todos los eventos, lo que facilita mucho el trabajo teniendo en cuenta que hablamos de marcar, colgar y descolgar 100 veces. Este tipo de pruebas sirven para descartar errores de conexión por congestión, interferencia, fallos en sistema radiante, y en definitiva todos

aquellos problemas que impiden que la llamada se inicie.

3. Descarga de datos. Las herramientas de Drive test permiten programar scripts para descargas de datos. Se pueden realizar descargas de ficheros vía FTP, realizar pruebas de navegación WEB, P2P, etc. automatizando todas estas descargas con un mínimo de supervisión.

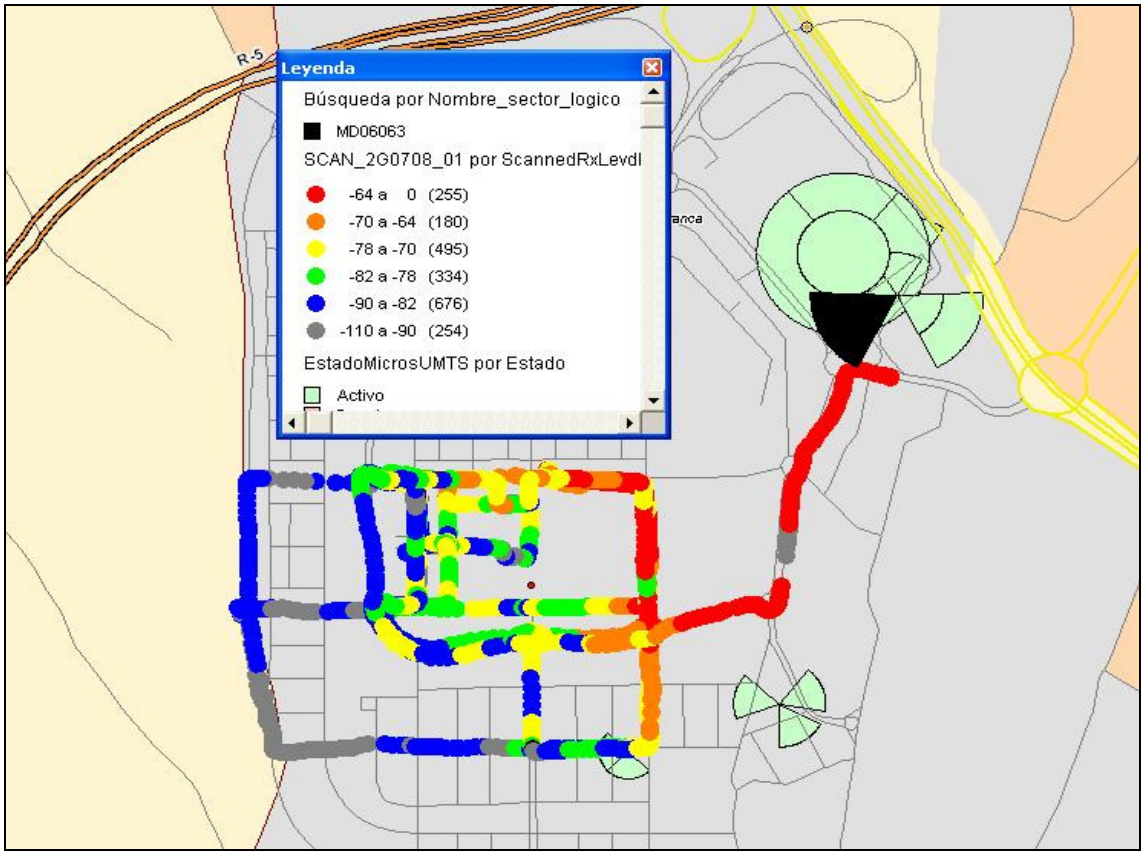

Figura 11: Niveles de señal capturados tras un Drive test

 Ventajas DT

- Las medidas realizadas van asociadas a un instante y punto geográfico determinado. Tenemos una foto precisa de los niveles de señal en localizaciones concretas.
- En un Log disponemos de todos los mensajes de capa 2 y 3 que se intercambian el UE y la red. De esta forma, ante una llamada caída se puede profundizar en el diálogo que mantenían ambas entidades segundos antes de producirse.
- Permite tomar muestras en interiores. Zonas más delicadas en cuanto a nivel de señal.
- Es la herramienta utilizada para realizar *Benchmarkings* entre operadores y conocer el estado de la competencia.
- Posibilidad de medida con scanner para capturar todas las frecuencias

Desventajas:

- La información recogida es puntual y normalmente bajo demanda. Es inabordable medir de esta manera el área geográfica de una red.
- Necesidad de contratar estos servicios a terceros. Coste elevado en material y RRHH: 2 técnicos por medida.

2.3. Sondas

Puede verse como la fusión de los dos métodos anteriores ya que aglutina parte de las ventajas de ambos.

Como su nombre indica, se trata de colocar un elemento capaz de leer todos los mensajes que atraviesan una determinada ruta en un interfaz (a nivel de BSC/RNC). Sería algo parecido a un analizador de protocolos en el que para cada comunicación podemos monitorizar los mensajes intercambiados entre UE y la red pero de forma agregada. El objetivo no es la localización exacta de los eventos sino la detección de comportamientos anómalos de red a nivel agregado.

La utilización de sondas basadas en monitorización de red se utilizan principalmente para:

- Detección de comportamientos inaceptables de red, permitiendo al operador tomar las acciones oportunas de optimización
- Detección de errores en los algoritmos de control de tráfico.
- Proporcionar a los grupos de planificación de red, información detallada acerca del dimensionamiento y configuración de la misma.

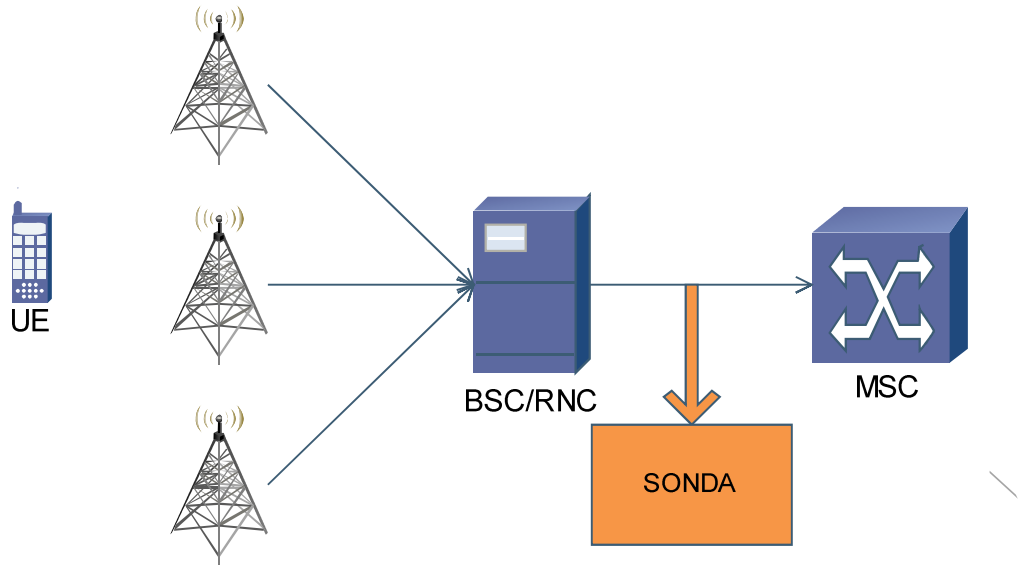

Figura 12: Esquema de monitorización por Sonda

 Ventajas Sondas

- Permite conocer el contexto concreto en que se producen los eventos, dado que estamos almacenando el escenario completo de la transmisión. Por ejemplo, conocer la frecuencia exacta en la que estaba enganchado el teléfono antes de caerse la llamada.
- Detección de vecinas no añadidas. Este es uno de los motivos por los que más se utiliza en la práctica este método: Si existe una vecina que debería entrar en la lista y no está definida, nos aporta información concreta de qué vecina es necesario añadir. En UMTS los móviles informan a la red de las estaciones que están monitorizando con buenos niveles de señal. Si estas estaciones no están en la lista de vecinas nunca entrarán en el Active Set (no cursarán la llamada) pero al disponer de sondas podemos capturar esos mensajes y decidir añadir estas estaciones como vecinas en acciones de optimización.

 Desventajas:

- Excesiva cantidad de información para almacenar. Se genera en poco tiempo mucha más información que en el método de generación de contadores.
- Para cada llamada la señalización intercambiada es muy elevada con lo que monitorizar 2000 llamadas al día (algo normal en una celda) puede acumular varios GBytes de información. Por eso suelen programarse grabaciones de pocas horas y contra elementos de red concretos.
- Carga de Procesador. Alineado con el problema anterior, un elemento de red utiliza sus recursos computacionales en la gestión y enrutamiento de las llamadas como tarea principal. Al programar grabaciones, las CPU de estos elementos tienen que dedicar parte de sus recursos a la monitorización y grabación de los mensajes requeridos, peligrando de esta forma el funcionamiento normal del elemento. Como ejemplo decir que una RNC que supera el 80 % de su carga computacional en hora cargada es un serio candidato a ser ampliado (mediante HW). Al utilizar sondas de manera masiva pueden alcanzarse estos valores con facilidad.
- Al igual que el método 1, la información es agregada a nivel de celda, site o BSC/RNC. No disponemos de las localizaciones exactas que disparan los eventos.

2.4. Medidas de cobertura basadas en reclamaciones del Cliente

Sin duda el método más directo de detección de fallos de la red o niveles de señal débiles.

Este método utiliza como fuente de información la comunicación directa con los usuarios del servicio y cuya experiencia se almacena en sistemas CRM para su posterior análisis. Tanto si el cliente llama al servicio de atención (SAC) como si es partícipe de una encuesta de satisfacción, disponemos de primera mano de la percepción de la calidad de servicio en ubicaciones específicas proporcionadas por el cliente.

Se puede realizar una catalogación de quejas basadas por ejemplo en tecnología (GSM o UMTS), voz o datos, municipio, Código postal o dirección exacta para localizar zonas sensibles de la red en que la cobertura o calidad de servicio está degradada.

 Ventajas Atención al cliente

- Permite conocer la ubicación exacta en la que se encuentra el problema de cobertura. Al comunicarse con atención al cliente podemos determinar el problema concreto y almacenar la dirección en que se produce.
- Idealmente es una fuente de información perfecta. Disponemos de la cartera completa de clientes a nuestro servicio. Y no solo para problemas de cobertura. Si el cliente nos informa que no puede navegar es posible que el nodo que le da servicio tenga un fallo SW, o si informa que escucha ruidos al hablar puede ser problema de una emisión ilegal de RF. Incluso si la cobertura del operador es muy mala en un municipio pero allí no tiene apenas clientes, el operador puede decidir si desplegar o no su red, en función de sus objetivos comerciales.

 Desventajas:

- Sólo un pequeño porcentaje de los clientes que se encuentran con problemas en el servicio finalmente formalizan una queja en el SAC. Para la mayoría de los clientes concluir una llamada de forma correcta 95 de 100 veces, les supone un servicio correcto, pero es una proporción malísima en términos de calidad.
- Ausencia de objetividad en las reclamaciones. En este aspecto influyen muchísimos factores pero el más importante es el que atañe a la parte emocional del individuo. El momento en el que un cliente llama insatisfecho por el servicio, el mensaje se ve interferido por el estado de ánimo en que se encuentra en ese momento. Aunque el problema se encuentre en su domicilio la queja puede extenderse a su lugar de trabajo, a su barrio. Incluso puede ser una reclamación ficticia para romper compromisos de permanencia o rescindir contratos.

- Credibilidad: Gran porcentaje de las reclamaciones no hacen referencia a la calidad de la red móvil. Aunque el cliente asocie rápidamente la degradación del servicio a problemas de cobertura, en muchas ocasiones el problema es realmente del terminal, de la tarjeta SIM, de facturación, y cada vez más de finalización del bono de máxima velocidad en tarifas de datos.
- Conocimientos no técnicos del cliente. Es otro de los factores que más distorsionan un mensaje de reclamación. Al cliente se le pregunta por el modelo de terminal, si el problema es en 2G ó 3G, desde cuándo le ocurre, si ha probado en otros teléfonos. Pero un gran parte de la población no conoce estas cuestiones planteadas que ayudan a acotar el problema, si éste fuera real. En este aspecto, en los últimos años se han visto incrementadas las quejas de uso de redes 3G en ordenadores personales, lo que añade un componente HW mucho más complejo y fuente de infinidad de errores.
- Por todo esto, la recolección de quejas de cliente suele ser relevante únicamente en el caso en que se registren varias reclamaciones diferentes sobre la misma zona. Esto hace suponer que existe un problema real en cuanto a la calidad radio en esas zonas.

En resumen, de los cuatro métodos presentados, el **objetivo** es explotar el relativo al Drive Test: **monitorización directa desde el terminal móvil** de los niveles de señal y eventos. Eso sí, con una **versión SW simplificada** que no requiera grandes esfuerzos de procesamiento y facilite su posterior análisis.

3. Diseño

Tras exponer los objetivos del proyecto e introducir brevemente las funciones básicas del mismo, llega el momento de describir en detalle el sistema, los elementos implicados y el comportamiento de cada uno de ellos.

El sistema consta de dos entidades bien diferenciadas: Terminal móvil + Servidor central. En el móvil (en adelante UE: *User Equipment*), se instala la aplicación encargada de ejecutar en segundo plano y totalmente trasparente al usuario, las funcionalidades de monitorización remota. La función principal es la de detectar eventos de llamada caída y almacenarlos localmente para posteriormente comunicarlos al servidor. Y de forma paralela, almacenar niveles de señal en cada posición geográfica dentro de la zona ordenada por el servidor.

Se trata por tanto de arquitectura Cliente/Servidor en la que una entidad centralizada es capaz de almacenar y gestionar información de varias fuentes distribuidas geográficamente y cuyo potencial radica en el agregado total de medidas.

La capacidad de proceso, donde reside la inteligencia del sistema, está repartida entre los móviles que se comunican en algún momento con el servidor central y que son independientes. Esta última idea es crucial: el servidor manipula medidas muy diversas de los terminales, independientemente de que sean de un fabricante u otro (diferente HW para poder arrojar resultados estadísticos consistentes).

En la sección 3.1 explicaremos las funcionalidades que debe tener el sistema con el fin de cumplir con los objetivos planteados originalmente en el proyecto. En dicha sección, se analizarán en detalle cada una de estas funcionalidades. En la sección 3.2 presentamos la arquitectura del sistema además de organizar en forma de bloques funcionales cada uno de los componentes del mismo. En el apartado 3.3 exponemos en forma de diagramas de flujo la secuencia de actividad del sistema y mediante cronogramas se presentan estas secuencias ordenadas cronológicamente en función de los eventos que disparan cada una de las acciones. Ya en la sección 3.4 explicaremos en detalle el formato de los ficheros generados en la captura de datos y cómo son procesados posteriormente para la presentación final de resultados.

3.1. Funcionalidades del Sistema

En este apartado se exponen las funcionalidades principales que ejecutan clientes y servidor:

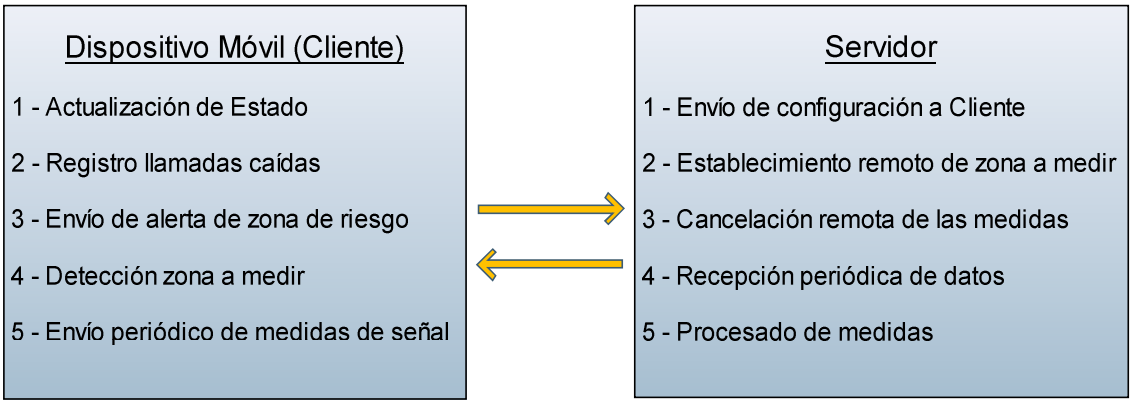

Figura 13: Funcionalidades del sistema mapeadas en Cliente y Servidor

Estas funcionalidades definen el comportamiento general de la aplicación y se exponen en detalle en los siguientes apartados. Resumiendo de forma general el comportamiento del sistema, tenemos por una parte el terminal (Cliente) que en todo momento mantiene actualizado el estado de teléfono almacenando los parámetros de estudio como veremos justo a continuación en 3.1.1. Además "escucha" el estado de cada llamada para almacenar en local todos los casos de llamada interrumpida y enviar una alarma al servidor central si detecta zonas de riesgo (3.1.2). Además es capaz de realizar medidas de cobertura en regiones ordenadas por el Servidor y almacenarlas en local (3.1.4) para posteriormente enviar los datos de forma periódica (3.1.5)

Por otro lado, desde el servidor central se implementa la configuración remota para inicializar la aplicación, configurar parámetros de la misma o incluso cancelar medidas (3.1.3). Además de forma centralizada es donde se recogen los datos de todos los dispositivos distribuidos que conforman el sistema para procesar las medidas (3.1.5). Veamos cada una de las partes a continuación:

3.1.1. Actualización del estado del teléfono

El servidor necesita toda la información posible en el momento en que sucede un evento destacable o en los instantes de medida de la zona de estudio. Por eso el terminal debe mantener actualizado el valor de los parámetros clave de la aplicación para registrarlos en instantes concretos. El teléfono utiliza todos estos parámetros para su funcionamiento habitual (medidas de intensidad para la conectividad con la red móvil, señal GPS para aplicaciones de localización, reloj del sistema, etc.) y para nuestros intereses queremos capturar estos valores desde el **Sistema Operativo** a través de librerías específicas de desarrollo y presentarlas a **nivel de aplicación**.

No necesitamos conocer todos los parámetros radio ni conocer en detalle el estado del terminal, pero para alcanzar el objetivo propuesto es imprescindible almacenar los siguientes **parámetros**:

1. **Tiempo**: En cada muestra se debe asociar un "*timestamp*". Recordemos que el objetivo último es la optimación de la calidad de red y si ha sucedido un evento excepcional es necesario conocer cuándo para asociar dicho evento a un estado de red concreto. Puede haber coincidido con un fallo puntual de una BTS (Base *Trasceiver Station*) por ejemplo.

2. **Posición**: Una vez obtenido el instante exacto de la entrada en el registro, el siguiente valor necesario es la posición. De hecho es la información clave que necesita saber el operador de red para localizar zonas especialmente sensibles.

3. **Sistema de posicionamiento**: En el proyecto trabajaremos con posicionamiento por GPS y basado en red, necesario conocer en cada momento el método empleado dado que la localización GPS es mucho más precisa. Lo veremos más adelante, en la sección de Implementación.

4. **Nivel de señal**: Registrando este valor en el instante de llamada caída podremos deducir si las condiciones radio eran buenas o no cuando se produjo el evento. Además en el "modo medida" del terminal, con los valores de señal podremos trazar un mapa de cobertura instantáneo en un área concreta.

5. **Estación Base** en la que está conectado el terminal: Es otro de los parámetros clave. La identificación de la estación se hace por medio de la dupla LAC/CID (*Local Area Code/Cell Id*). Si se detecta un número anormal de desconexiones en una celda concreta gracias a la aplicación, es fácil acotar el problema no sólo en una zona geográfica, sino en el elemento de red concreto que está causando el fallo.

6. **Estado de la llamada:** Identifica el estado de actividad del teléfono con respecto a la red. Identificamos 3 estados: *Idle* (reposo), *Ringing* (llamando) y *Active* (hablando). De esta forma podremos diferenciar comportamientos en reposo y en llamada.

7. **Modo de Red: GSM/UMTS.** La aplicación conoce en cada momento si el terminal está conectado a la red 2G ó 3G del operador. Implica diferentes maneras de abordar el análisis de datos.

8. **IMEI (*International Mobile Equipment Identity*):** Otro de los valores que siempre conoce la aplicación es el IMEI, que permite identificar el modelo del terminal. Esta información también se envía al servidor y con ella se pueden detectar tendencias de comportamiento o sensibilidad del HW radio en determinados modelos o fabricantes.

9. **Número de teléfono**: Es el identificador del subscriptor o cliente. No es especialmente crítico pero aporta además indicadores personalizados por cliente. Si un usuario presenta una tasa de llamadas fallidas muy elevada, parece interesante investigar por qué.

10. **Operador de Red**: Otro de los parámetros radio que puede capturarse es el de operador de red. Es un Identificador unívoco internacional que se incluye en los mensajes que intercambian UE y Red y que puede obtenerse desde el sistema. Es información redundante si la entidad que procesa los datos recogidos es el propio operador pero el sistema puede montarse en la sede de un fabricante por ejemplo. De esta manera, pueden estudiar el comportamiento de sus equipos en distintas redes móviles y en diferentes escenarios.

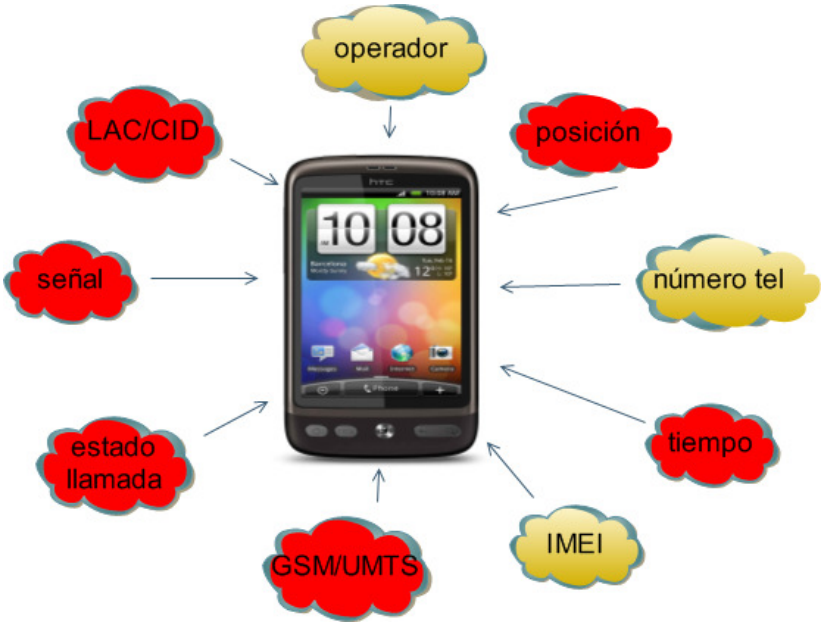

Figura 14: Parámetros capturados por la aplicación

En la figura 14, aparecen en rojo los valores que deben actualizarse en el registro de la aplicación cada vez que cambian. Por el contrario los parámetros como IMEI o el operador se almacenan una vez en el arranque.

3.1.2. Detección de llamada caída y zona de riesgo

Es una funcionalidad clave del proyecto. El programa cliente, es capaz de detectar las llamadas de voz que no terminan de forma normal sino que un problema excepcional ha provocado el corte de la misma. En el instante en que sucede, el sistema almacena el estado en el que se produce con todos los parámetros del

apartado anterior que aportan un contexto "semántico" a la llamada caída: dónde se produce, cuándo, en qué tecnología, etc. Esta información no se transmite directamente al Servidor, dado que la interrupción de llamadas es algo relativamente normal en el uso diario del servicio de telefonía y una caída aislada a priori no es relevante.

El conjunto de llamadas caídas (junto con las medidas de cobertura que veremos más adelante) se almacenan en local para transmitirlas periódicamente hacia el servidor en una hora concreta del día (o de la semana, a configurar por el Servidor)

Pero existe un escenario que implica **la comunicación inmediata** con el servidor central: los casos en que se producen llamadas caídas muy seguidas en tiempo o en espacio. Así, se identifican dos posibles casos:

1. Dos llamadas caídas en menos de T segundos:

2. Dos llamadas caídas en menos de M metros.

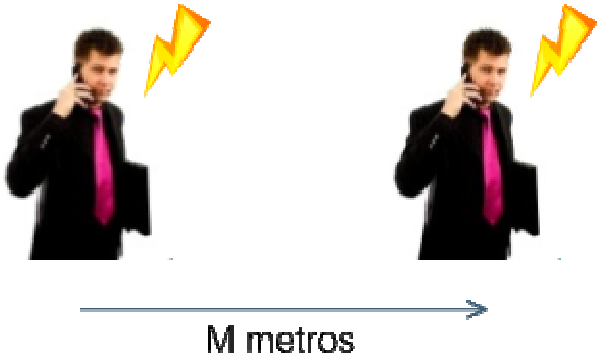

Figura 15: Detección de zona de riesgo

En cualquiera de estos casos, el terminal envía la información de Estado al servidor de forma inmediata porque se trata de situaciones excepcionalmente relevantes para atacar posibles fallos de red lo antes posible. Es responsabilidad del operador definir la resolución de estas alarmas para evitar falsos avisos. Así, se pueden configurar en remoto el tiempo y distancia de las alarmas así como el número de llamadas caídas que se consideran "excepcionales".

3.1.3. Configuración Remota desde el Servidor

El comportamiento del terminal en la captura de medidas es regido directamente por el servidor, que de forma remota es capaz de:

1. **Activar la aplicación** en el móvil si esta se encuentra apagada. El único requisito para la captura remota de medidas es que la aplicación esté instalada. Para arrancar dicha aplicación pueden darse 3 escenarios:

   a) Que se ejecute automáticamente al encender el móvil.
   b) Que arranque la aplicación el usuario de forma manual.
   c) Y para no dejar el control únicamente al usuario, desde el servidor pueden enviarse órdenes al sistema operativo para iniciar la aplicación por si esta no estuviera activa.

2. **Configuración de medida**. Es este aspecto el que confiere realmente inteligencia al sistema y con mayor contenido semántico en la comunicación Cliente/Servidor. Con un mensaje concreto se comunica al UE una **determinada zona geográfica** a medir, enviando en el mensaje de configuración **centro y radio** de una circunferencia que contiene la zona de estudio (medida de cobertura). Se explica con más detalle en la sección siguiente, 3.1.4.

3. **Cancelación Medida**. Desde el servidor, con un mensaje sencillo. puede cancelarse la funcionalidad de medida de la aplicación.

En la sección 3.3 dedicada a presentar los diagramas de flujo y cronogramas puede entenderse **de forma general** qué eventos tienen lugar en el funcionamiento natural del sistema y que acciones desembocan.

3.1.4. Detección de zona de medida de cobertura

Cuando el dispositivo está dentro de la zona configurada desde el servidor, comienza a almacenar medidas de señal (junto con la posición, tiempo, y el resto de parámetros vistos en el apartado 3.1.1) y en el momento que abandona la región, se detiene el registro de medidas. De manera gráfica, el almacenaje de medidas de señal en zonas específicas, puede verse en el siguiente dibujo:

Figura 16: Ejemplo de medida de cobertura en "*red zone*"

Donde representamos con colores cálidos las zonas de mejor cobertura (señal más potente detectada por el UE)

Los motivos de configurar zonas de medidas concretas y no medir constantemente (en cualquier lugar) son los siguientes:

- Ahorro de batería: Ejecutar la aplicación de manera continua consumiría recursos energéticos de forma constante en un dispositivo cuya optimización del consumo es clave.
- Almacenamiento local: Los terminales móviles, aunque cada vez más avanzados, poseen limitaciones en memoria y capacidad de almacenamiento. Tomar muestras las 24h del día generaría ficheros de varios Mb, lo que acabaría rápidamente con la memoria interna del dispositivo, siendo necesaria tarjetas de expansión.
- Transferencia de datos: Este es quizá el factor más importante en la limitación de zonas a medir. A la hora de transmitir los datos al servidor central, se ocupan recursos de red y el conjunto de dispositivos activos del sistema de medida puede ser muy grande. Por eso es interesante que la información transmitida no sea muy pesada. No tiene sentido contribuir a la congestión de una red cuyo objetivo es optimizar.

Por estas razones, se establece además en la aplicación un número máximo de muestras consecutivas a medir. Si por ejemplo la aplicación se instala en un usuario que circula durante todo un día por Leganés y se configura el municipio completo como zona de medida, el resultado del registro sería muy voluminoso. Por eso, se limita también en número de muestras.

Pero… ¿Qué zonas medir?, Es responsabilidad absoluta del Servidor central. Se definen áreas específicas de medida a comunicar a los terminales por 2 motivos:

- Área de especial interés para el operador por ser zonas sensibles de calidad radio. Detectadas por estadísticas de red, reclamaciones de clientes o cualquiera de los métodos vistos en el apartado de Estado del Arte.

- Área en el que se detecta zona de riesgo por parte de los móviles, tal como vimos en el apartado 3.1.2.

3.1.5. Envío de Información y procesado de datos

Tanto las llamadas caídas detectadas como las medidas de cobertura registradas en las zonas de estudio se almacenan de forma local en el dispositivo móvil en forma de ficheros de texto con formato específico. Esta información ha de trasladarse a la unidad central de procesado que recoge todos los eventos y medidas registrados de cada uno de los dispositivos que componen el sistema global. La comunicación se realiza estableciendo una comunicación GPRS en caso de permanecer en GSM y mediante comunicación de datos HSDPA si el terminal se encuentra en la red 3G.

La transferencia se realiza en dos instantes de tiempo diferentes atendiendo a la urgencia en la comunicación con el Servidor:

1. Comunicación Urgente. En el instante en que se detecta una zona de riesgo por parte del móvil, se realiza la transferencia de los ficheros generados.

2. Comunicación periódica. Es el modo normal de funcionamiento: Durante el día el terminal registra llamadas caídas y medidas de cobertura en zonas concretas. Siempre que los ficheros no estén vacíos (es posible que no se generen eventos algunos días) se realiza la comunicación con el servidor de forma periódica en una hora concreta, eligiendo como hora de conexión aquellos momentos de baja carga de tráfico de datos en la red. Por ejemplo: Transferencia diaria a las 4 am.

Una vez que el operador de red dispone de los ficheros de todos los clientes que componen el sistema, las medidas se procesan con herramientas de cartografía para detectar zonas especialmente sensibles en eventos de llamada caída o de problemas de cobertura ya sea por fallo de red o de falta de despliegue en alguna tecnología. Se explica en detalle en la sección 3.4.

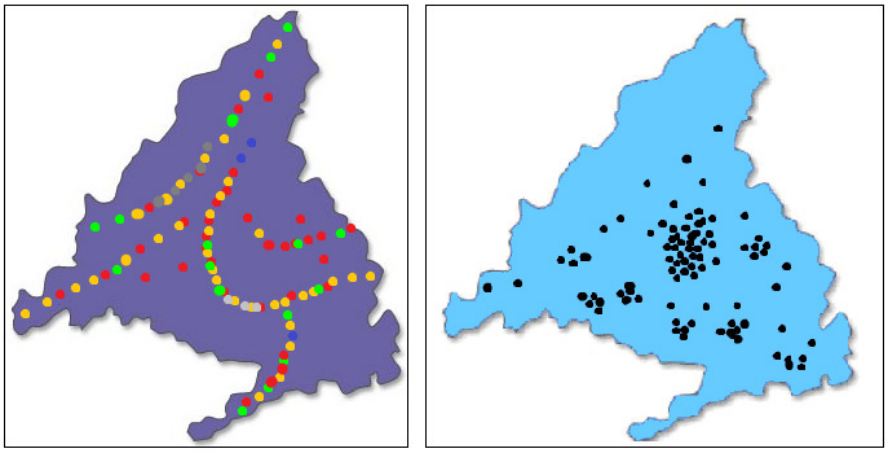

Figura 17: Ejemplo de procesados de muestras

3.2. Arquitectura y bloques del sistema de monitorización

Ahora que se han presentado las funcionalidades principales del sistema llega la hora de ubicar cada uno de los bloques que realizan dichas actividades y como están relacionados. En primer lugar representamos de manera gráfica la manera en que está constituido el sistema completo en forma de **bloques** (3.2.1). Posteriormente, en la sección 3.2.2, se define la arquitectura, que se centra principalmente en la **comunicación** entre los distintos elementos.

3.2.1. Diagrama de Bloques

En esta sección se definen los bloques semánticos que componen el sistema (tanto en cliente como en servidor), encargados de instanciar las funcionalidades presentadas en la sección 3.1, además de la relación que existe entre cada uno de los elementos. En la siguiente figura, se ilustran dichos elementos y cómo se comunican entre sí:

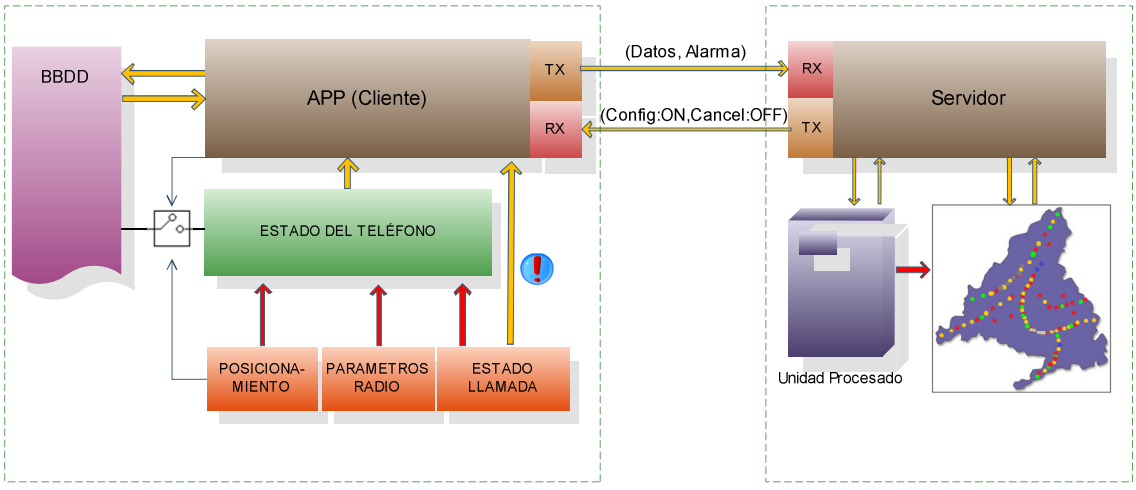

Figura 18: Diagrama de bloques del sistema

El bloque clave en la aplicación cliente es el “Estado del Teléfono”. Debe estar actualizado cada vez que alguno de los valores que contiene cambia, especialmente críticos son la posición y nivel de señal (dentro de los parámetros radio).

Toda esta información de estado se almacena en memoria y el siguiente estado sobrescribe al anterior. Únicamente se registra el estado en fichero de texto en los instantes en que sea necesario. Por eso en la figura se representa un conmutador

controlado por dos entidades: Posición y Servidor (a través de mensajes de configuración).

Si la aplicación corriera en modo libre, eliminando el conmutador de la figura, cada actualización de estado sería registrada en disco (fichero) y cómo vimos en el apartado 3.1.3, esto ni es rentable, ni es el objetivo del sistema de monitorización.

Únicamente se registra la información de estado en disco cuando nos encontremos dentro del área específica de medida. En esos casos el bloque de geolocalización, dispara *triggers* específicos al sistema para indicar que se está dentro de la región de estudio, permitiendo el registro de la medida de cobertura. Por otro lado, si el servidor central cancela las medidas de los dispositivos, tampoco se almacenaría en fichero ninguna medida.

El registro de llamada caída es totalmente diferente, **siempre** que se detecte una llamada interrumpida de forma anormal, se registra en fichero de salida, sin restricciones. Además, como puede verse en la figura, se implementan mecanismos de alarma, una vez que se detectan situaciones potenciales de riesgo como hemos visto antes. No sólo se registra la caída en disco sino que inmediatamente se comunica al servidor central los eventos y el estado en qué se produjeron y se traslada la responsabilidad de análisis a la parte de procesado de medida.

Los 3 elementos que actualizan el estado del teléfono son:

1. POSICIONAMIENTO

   Es una de las fuentes de información más importantes del sistema. El UE debe conocer en cada momento dónde se encuentra y por eso almacena en memoria latitud y longitud, actualizando estos dos valores cada vez que se ven alterados.

   Hace pocos años la inclusión de módulos GPS en los terminales era impensable para el mercado de consumo pero actualmente raro es el *smartphone* que no viene equipado con él y además a precios asequibles. Es una gran ventaja no sólo como información de posicionamiento para el usuario sino para todas las aplicaciones que se basan en localización: *Latitude* de Google, *Foursquare*, por poner algún ejemplo y por supuesto la que presentamos en este Proyecto.

   Además, la obtención de posición, no es exclusiva del sistema GPS. En la actualidad casi todos los terminales permiten almacenar la posición mediante varios métodos, cada uno con determinadas características sobretodo de consumo de batería y de precisión en la localización. Destacamos las más habituales:

   a) **GPS**: (*Global Positioning System*)

El posicionamiento mediante GPS, se basa en medidas de retardo de señal entre al menos 3 satélites que orbitan alrededor de la Tierra de entre los que componen la constelación total, a unos 20.000 Km. de la superficie y que permite establecer la posición a nivel terrestre con sólo algunos metros de error. El requisito es disponer de HW específico para captar dichas señales, Handicap que ha superado la tecnología actual con creces en cuanto a precios y tamaño de integración. Como ejemplo, en la figura se muestra la antena GPS del Iphone:

Figura 19: Antena GPS Iphone

Es la mejor solución en cuanto a precisión de localización y no requiere conexión de datos por lo que no repercute en coste extra. El problema del sistema GPS es el consumo de batería, recurso muy escaso en dispositivos portátiles.

b) **A-GPS**: (*Assisted GPS*)

Apoyo a GPS mediante la red celular basado en servidores de localización. Entre sus principales ventajas está que disminuye la conexión en frío a pocos segundos comparado con minutos del GPS convencional (es el tiempo de la primera conexión GPS tras mucho tiempo de inactividad). Conlleva coste de datos para obtener esta información pero cada conexión no supera los 10 KB. En cualquier caso, el uso en *roaming* (fuera de la red doméstica) al viajar a otros países incrementa este coste de manera notable.

c) **Basado en red**:

No requiere ni HW especial ni conlleva gastos en conexiones de datos ya que se basa en métodos de triangulación de celdas. A partir del CGI (*Cell Global Identity*: Identificador Internacional unívoco de estación móvil) de varias celdas que se monitorizan a la vez, el terminal es capaz de calcular retardos y niveles de potencia de las señales que radian, para calcular de manera aproximada la ubicación del UE. La resolución es mucho

peor que un GPS (varias decenas de metros) pero permite ubicar el dispositivo incluso en entornos dónde no llega la señal GPS (Túneles, parkings, etc.). El consumo energético es escaso ya que utiliza medidas que los terminales ya están utilizando para la comunicación celular normal.

El sistema debe ser capaz de chequear por orden de preferencia qué método de posicionamiento está activo. La autorización de localización en el UE forma parte del perfil del usuario y es posible que ninguno de los métodos esté disponible. En ese caso se devuelve un mensaje de error.

Lo normal es que la aplicación comience por chequear si el GPS está activado y continuar con los siguientes métodos hasta obtener la posición (normalmente mayor prioridad al GPS)

Una vez obtenida ésta, para su manipulación únicamente se necesitan dos variables decimales que representan latitud y longitud (no vamos a considerar otros parámetros GPS cómo velocidad o altitud). Con la posición actualizada, el bloque de localización es el encargado de comparar la posición actual con el área de medida configurada por el servidor. Si se encuentra dentro de dicha región comunica al hilo principal una señal para autorizarle a almacenar medidas en disco. El caso contrario se da cuando el UE abandona una zona de medida.

En el caso de estar en el interior de una zona de medida, se actualiza el estado de los parámetros radio cada vez que cambia la posición, aunque no haya cambios en señal, celda servidora, etc. Para evitar medidas redundantes, se define un umbral de distancia dentro del cual el bloque de posicionamiento no actualiza el estado a la capa de aplicación. Tener una medida cada 50 cm puede no ser relevante y añadiría información redundante, por eso se establece una “distancia mínima de refresco”, por ejemplo 10 metros.

2. PARÁMETROS RADIO

Los sistemas operativos móviles disponen de librerías para desarrolladores que permiten obtener algunos de los parámetros que intercambian UE y red para el normal funcionamiento de la comunicación móvil, ya sea GSM o UMTS. Como hemos explicado en apartados anteriores, cada vez que cambia alguno de los parámetros que requieren actualización inmediata, es necesaria la comunicación asíncrona con el hilo principal del programa para que almacene en memoria siempre el estado más actualizado. Traduciéndolo a código, esta comunicación asíncrona de procesos en la aplicación cliente, se establece mediante “listeners”. Son elementos de código específicos que reaccionan a ciertos eventos. Se inicializan para reaccionar a determinadas señales e ignorar aquellas para las que no están programadas. En el otro lado, el código fuente debe incluir qué mecanismos disparan

determinadas señales o "*triggers*". Por ejemplo: Cuando el móvil detecta un cambio de nivel de señal en la recepción, se deben definir en el código dos bloques duales: El encargado de detectar dicho cambio y lanzar una señal específica (en nuestro caso lo hace el sistema operativo) y el *listener* diseñado específicamente para esa señal, que ejecuta la porción de código que actualiza las variables locales de nivel de señal (y por tanto el estado del teléfono sobrescribe el nuevo valor)

3. LLAMADA CAÍDA

Al igual que par la captura de parámetros radio, se define en el código un *listener* para llamadas interrumpidas y se almacena el estado: *Caída 1 Posición X Tiempo T*, acto seguido se registra el evento en disco con el resto de parámetros pero la información de la Caída 1 continúa en memoria. Cuando se produce una segunda caída se registra también en memoria *Caída 2 Posición Y tiempo Z*. Tras almacenar la caída se comprueba si se cumplen los requisitos de alarma (las dos caídas en una distancia o tiempo por debajo del umbral definido). En ese caso, además del registro se construye el mensaje de alarma y es enviado al servidor central.

En los siguientes apartados veremos el diseño de la arquitectura del sistema y los diagramas de flujo de las actividades que conforman el comportamiento natural del mismo.

3.2.2. Arquitectura

En la siguiente figura podemos ver una representación de la arquitectura del sistema en la que se identifican las entidades que intervienen en la comunicación y qué elementos de red son necesarios para trasladar la información de un punto a otro.

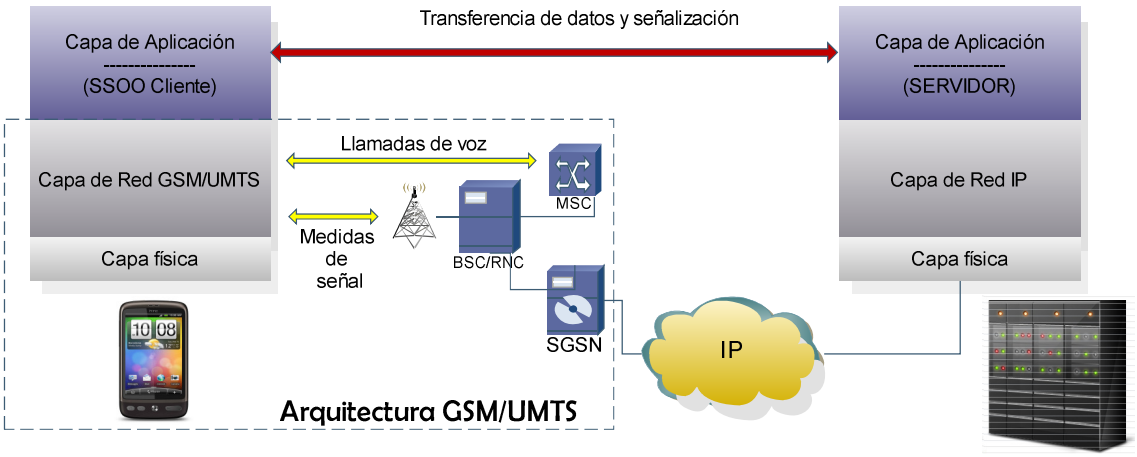

Figura 20: Arquitectura del sistema de monitorización

Desde el lado del UE, en la figura vemos un modelo simplificado de la arquitectura de la red móvil. El terminal, independientemente de las aplicaciones que alberga a nivel de usuario, gestiona directamente con la red todos los mensajes necesarios para el normal funcionamiento de la comunicación celular, ya sea GSM o UMTS. Nosotros aprovechamos parte de esta información para nutrir a la aplicación cliente de los datos necesarios que componen el mapa de puntos (cobertura y eventos) que posteriormente será transmitido al servidor central.

En el momento en que Cliente y Servidor han de comunicarse, es necesario establecer sesión de datos en el móvil. La transferencia de los ficheros se realiza mediante los elementos de red que comunican la red móvil con Internet. Es el plano de datos de la arquitectura GSM/UMTS la que gestiona dicha comunicación mediante SGSN (y GGSN).

En el ejemplo se representa un modelo único para GSM y UMTS, que comparten elementos en red para la gestión de datos. El SGSN (*Serving GPRS Support Node*) es el responsable de la gestión de paquetes de datos y que a través de GGSN (*Gateway SGSN*) conecta la red móvil con la red IP (Internet). El servidor, por tanto, es capaz de comunicarse con el terminal móvil a través de esta red IP y recibir los ficheros de manera sencilla vía FTP.

Desde el comienzo del Proyecto hemos planteado como gestor del sistema al propio operador de red, dado que es el propietario de los elementos de red por los que se transmiten tanto la voz como los datos. Pero en la arquitectura propuesta se define el servidor **fuera** de la red móvil. Es decir, no sólo puede instalarse la aplicación en un proyecto de gestión interno sino que puede externalizarse.

Cliente y Servidor “hablan” mediante Internet por lo que es independiente sobre qué medio físico viaja la información. Tanto la captura de datos como su transferencia es responsabilidad de las capas de aplicación. Este concepto es importante porque permite trasladar la capacidad de captura y análisis a entidades externas como fabricantes de dispositivos o consultoras independientes.

Los fabricantes pueden querer capturar información del comportamiento de sus terminales en calidad de señal, volumen de datos, etc. Por supuesto añadiendo algún tipo de reclamo para el usuario que tiene instalada la aplicación, normalmente económico.

Por otro lado, instituciones independientes como la CMT (Comisión del Mercado de Telecomunicaciones) por poner un ejemplo puede gestionar un sistema de este tipo para realizar estudios de comportamiento en el uso del servicio móvil en la población. También puede el operador externalizar estos estudios a Consultoras externas, para los que el modelo propuesto facilita su gestión.

3.3. Diagramas de Flujo y cronogramas

En los diagramas de flujo podemos ver de un solo vistazo la secuencia de pasos y actividades que se ejecutan en la aplicación. En la siguiente figura se muestra el diagrama de flujo del sistema desde el comienzo (antes de entrar en ejecución):

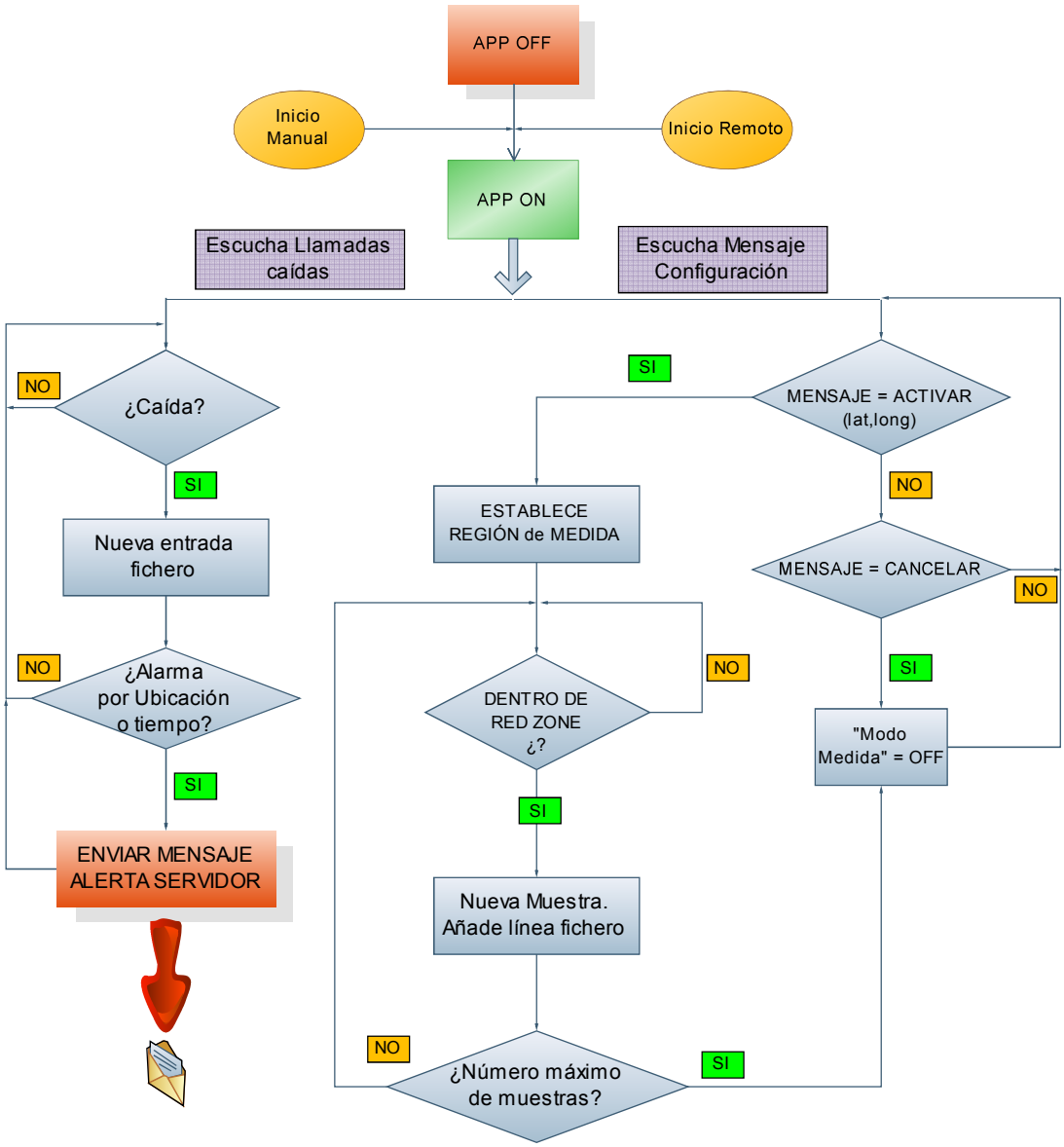

Figura 21: Diagrama de flujo de la aplicación

Al comienzo se representan las dos formas de inicializar la aplicación, manual y remota, para pasar directamente a la "escucha de eventos". Esta zona se divide en dos ramas, ya que el sistema reacciona a dos estímulos de forma paralela: Registrar las llamadas caídas y tomar medidas de cobertura en las zonas indicadas por el servidor.

En la primera rama, el programa detecta llamadas caídas para posteriormente almacenarlas en disco y contrastar si existe otra llamada caída en memoria que coincida en área o tiempo con la actual. De ser así se envía mensaje de alarma al servidor.

En la segunda rama se representan las acciones de medida de cobertura una vez se recibe el mensaje con el área a medir (denominada "Red Zone"). Si se recibe un mensaje de configuración se establece en memoria la posición y el radio de la Red Zone y en el momento en que se encuentra en su interior el programa comienza a almacenar muestras. Se detiene una vez que ha salido del área de medida o ha alcanzado el número máximo de muestras. En ese caso vuelve al estado inicial y espera mensaje de activación. Es el mismo estado de reposo al que se llega si se recibe un mensaje de "Cancelación" de la medida.

La comunicación Cliente Servidor y las acciones que se disparan ante determinados eventos, pueden representarse también mediante cronogramas. Veamos todos los casos:

a) Activación y detección de caída:

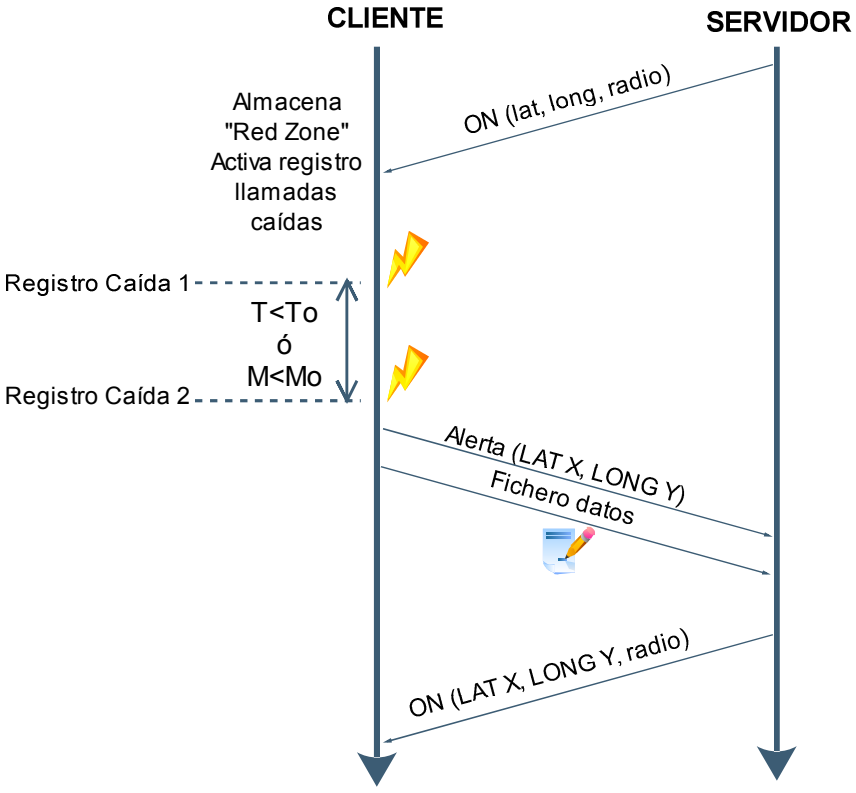

Figura 22: Cronograma Activación de sistema y captura de llamada caída

Desde el instante en que se recibe el mensaje de configuración, la aplicación arranca (si no lo estaba) esperando a que se produzca evento de llamada caída. Se registran estos eventos y en caso de incurrir en una nueva llamada caída dentro de los umbrales establecidos, se lanza mensaje de alerta al servidor y el fichero de eventos

almacenado. En ese momento el servidor genera una nueva orden de medida estableciendo como "*red zone*" la región donde se ha producido la última caída. Resultará interesante conocer las medidas de señal de la zona en que se ha dado el fallo.

b) Activación y medida de cobertura (2 posibilidades):

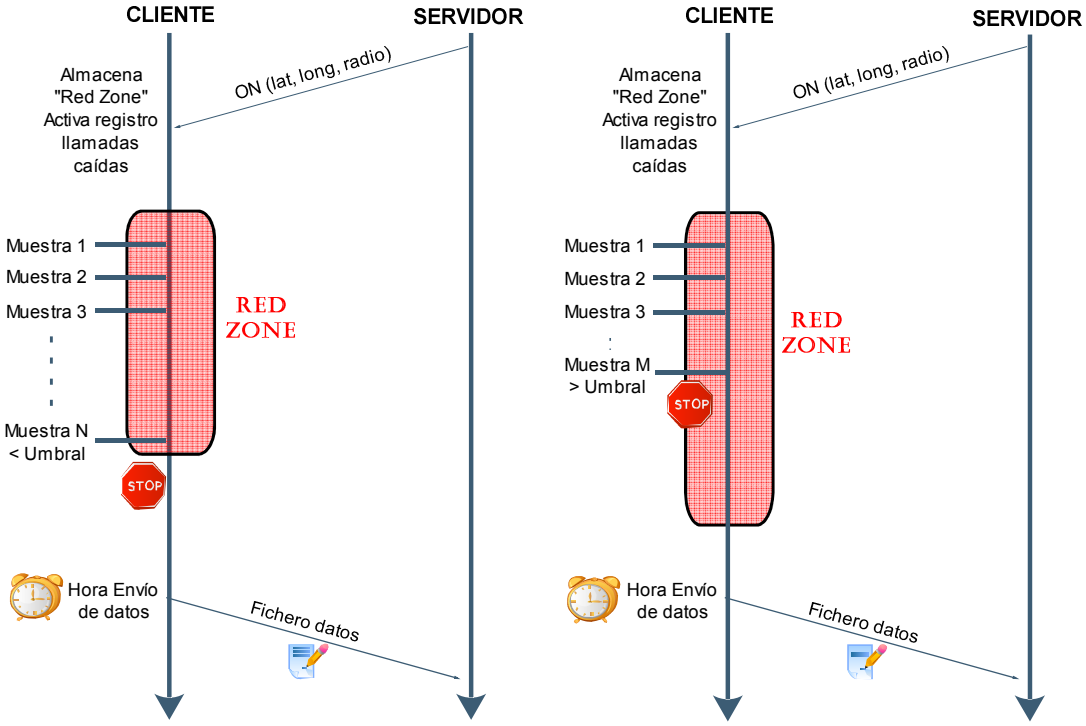

Figura 23: Medida en "red zone" con finalización por salida de la misma (izquierda) o por alcanzar número máximo de muestras (derecha)

En cuanto a las medidas de señal de zona de estudio, el terminal almacena en primer lugar el centro y radio en memoria. Al atravesar la región determinada comienza a almacenar muestras. En los diagramas se representan los dos escenarios: finalizar la captura de medidas al salir de la región (izquierda) o al llegar al número máximo de muestras (derecha). En ambos casos no se envía fichero de datos hasta llegar a la hora establecida para ello.

A continuación, el **resumen de eventos y acciones** del sistema, representando con una **(C)** las acciones del terminal (Cliente) y con **(S)** el Servidor.

1. **(S)** Envío de mensaje de configuración → **(C)** Almacenar en memoria área de medida e inicializar aplicación.
2. **Llamada caída** → **(C)** Almacenar en fichero información de la caída.
3. **Llamada caída bis** → **(C)** Almacenar en fichero la caída + envío de mensaje de alarma al servidor central → **(S)** Servidor envía nueva de zona de medida con centro la posición de la última caída.
4. **Entrada en zona de medida** → **(C)** Comienza la toma de muestras de cobertura.

5. **Número máximo de muestras capturadas → (C)** Fin de la captura de muestras.
6. **Salida de la zona de medida→ (C)** Fin de la captura de muestras.
7. **(S)** Envío de mensaje de cancelación de la medida → **(C)** Fin de la monitorización.
8. **Hora de envío periódico de medidas → (C)** Terminal inicia conexión de datos para enviar todos los ficheros de texto almacenados durante el día.

3.4. Procesado de resultados

Hasta este punto, se ha presentado la forma en que se capturan los datos y cómo son almacenados y transferidos a un servidor que contiene toda la inteligencia de análisis y procesado de resultados. La responsabilidad de aportar valor a toda la información obtenida por el parque de terminales asociados al sistema está centralizada en el dicho servidor.

Es el momento de presentar el formato de los ficheros generados por los UEs del sistema y cuyo conjunto aporta una base de datos de cierta envergadura para el posterior análisis centralizado. A continuación podemos ver una tabla ejemplo con las entradas de fichero que generaría un móvil con la aplicación instalada, donde se consideran una serie de muestras de medidas de señal y un evento de llamada caída:

Time	lat	long	num	IMEI	GSM - UMTS	Signal	State	GEO	LAC	CID	ISP	ACTION
16/06/2011 16:04:25	40,325	-3,458	651987654	4402140034996	GSM	-81	IDLE	GPS	1004	1892	Orange	medida
16/06/2011 16:04:29	40,325	-3,450	651987654	4402140034996	GSM	-81	IDLE	GPS	1004	1892	Orange	medida
16/06/2011 16:04:33	40,325	-3,442	651987654	4402140034996	GSM	-81	IDLE	GPS	1004	1892	Orange	medida
16/06/2011 16:04:37	40,325	-3,434	651987654	4402140034996	GSM	-81	IDLE	GPS	1004	1892	Orange	medida
16/06/2011 16:04:41	40,325	-3,426	651987654	4402140034996	GSM	-81	IDLE	GPS	1004	1892	Orange	medida
16/06/2011 16:04:45	40,325	-3,418	651987654	4402140034996	GSM	-84	IDLE	GPS	1004	321	Orange	medida
16/06/2011 16:04:49	40,325	-3,410	651987654	4402140034996	GSM	-84	IDLE	GPS	1004	321	Orange	medida
16/06/2011 16:04:53	40,325	-3,402	651987654	4402140034996	GSM	-84	IDLE	GPS	1004	321	Orange	medida
16/06/2011 16:04:57	40,325	-3,394	651987654	4402140034996	GSM	-84	IDLE	GPS	1004	321	Orange	medida
16/06/2011 16:05:01	40,325	-3,386	651987654	4402140034996	GSM	-84	IDLE	GPS	1004	321	Orange	medida
16/06/2011 16:05:05	40,325	-3,378	651987654	4402140034996	GSM	-84	IDLE	GPS	1004	321	Orange	medida
16/06/2011 16:05:09	40,325	-3,370	651987654	4402140034996	GSM	-82	ACTIVE	GPS	1004	42015	Orange	medida
16/06/2011 16:47:13	40,325	-3,362	651987654	4402140034996	GSM	-82	IDLE	GPS	1004	42015	Orange	CAÍDA
16/06/2011 16:05:17	40,325	-3,354	651987654	4402140034996	GSM	-82	IDLE	GPS	1004	42015	Orange	medida

Figura 24: Formato de fichero de texto generado por el UE

El fichero generado está compuesto por diferentes líneas de texto en ASCII, donde cada una de ellas representa una muestra tomada por el móvil (si se encuentra

en zona de medida). En cuanto al registro de las llamadas caídas, el formato es idéntico, se acompaña a la posición del evento con el resto de parámetros que hacen más comprensible el contexto de la llamada.

Una vez recopilados todos los ficheros, el objetivo es analizarlos y desgranar por partes toda la información que contienen en estudios de cobertura, de eventos, por fabricante, por operador, etc.

Para poder representar gráficamente los puntos geográficos que contiene cada línea del fichero y poder realizar análisis más o menos complejos de los datos, vamos a utilizar herramientas de procesado GIS (*Geographic Information System*).

Por definición, un **sistema GIS** es una integración organizada de hardware, software y datos geográficos diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión. Para ello existen en el mercado diferentes soluciones SW.

Permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones en forma de capas superpuestas. Es un conjunto de herramientas que resulta imprescindible para nuestro análisis dado que desde el comienzo, la base del análisis propuesto es cartográfica.

Mediante herramientas GIS, seremos capaces de representar de manera gráfica y sencilla la posición geográfica concreta da cada uno de los eventos y valernos de funcionalidades de representación temática para construir mapas de cobertura. Esta representación temática asigna un diseño específico (colores o tamaños) a determinados rangos de cobertura, arrojando resultados como los que se muestran a continuación:

3.4.1. Análisis de cobertura

Es uno de los principales tipos de análisis que pueden hacerse con las medidas capturadas. Para cada punto geográfico por el que pasa el terminal en "modo medida", asociamos un nivel de señal que en el procesado se presenta cartográficamente con colores en función de la intensidad de la señal recibida. La figura siguiente es un ejemplo donde los colores cálidos indican mejor cobertura:

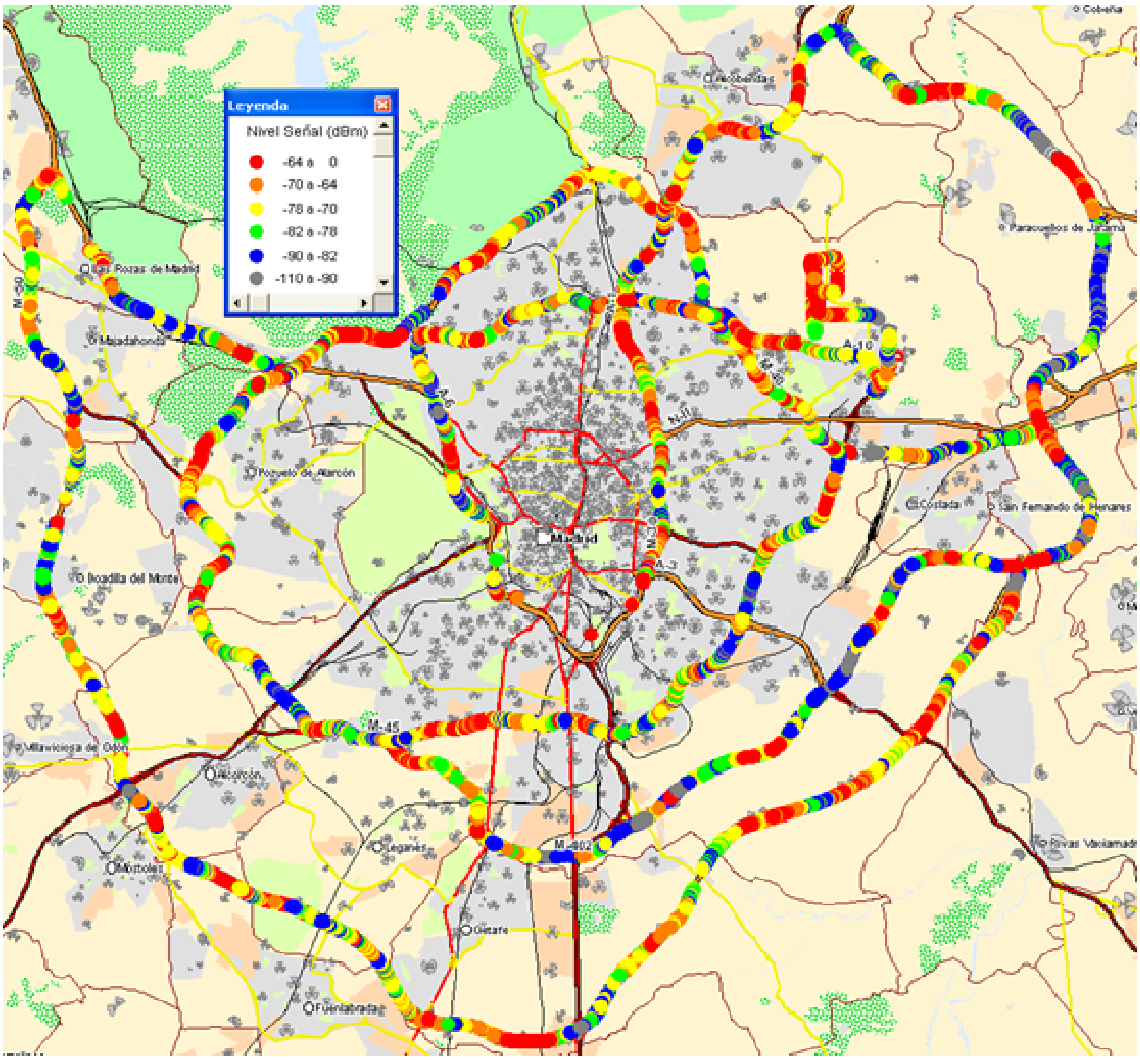

Figura 25: Ejemplo de mapa de cobertura generado por captura de muestras desde dispositivo móvil

3.4.2. Análisis de llamadas caídas

El segundo tipo de análisis 'tipo' es el de localización de llamadas caídas. Una vez recogidos todos estos eventos de todos los terminales que aportan muestras al servidor central, puede representarse el conjunto de puntos para detectar zonas con alta densidad de llamadas caídas que pueden indicar zonas con problemas de servicio:

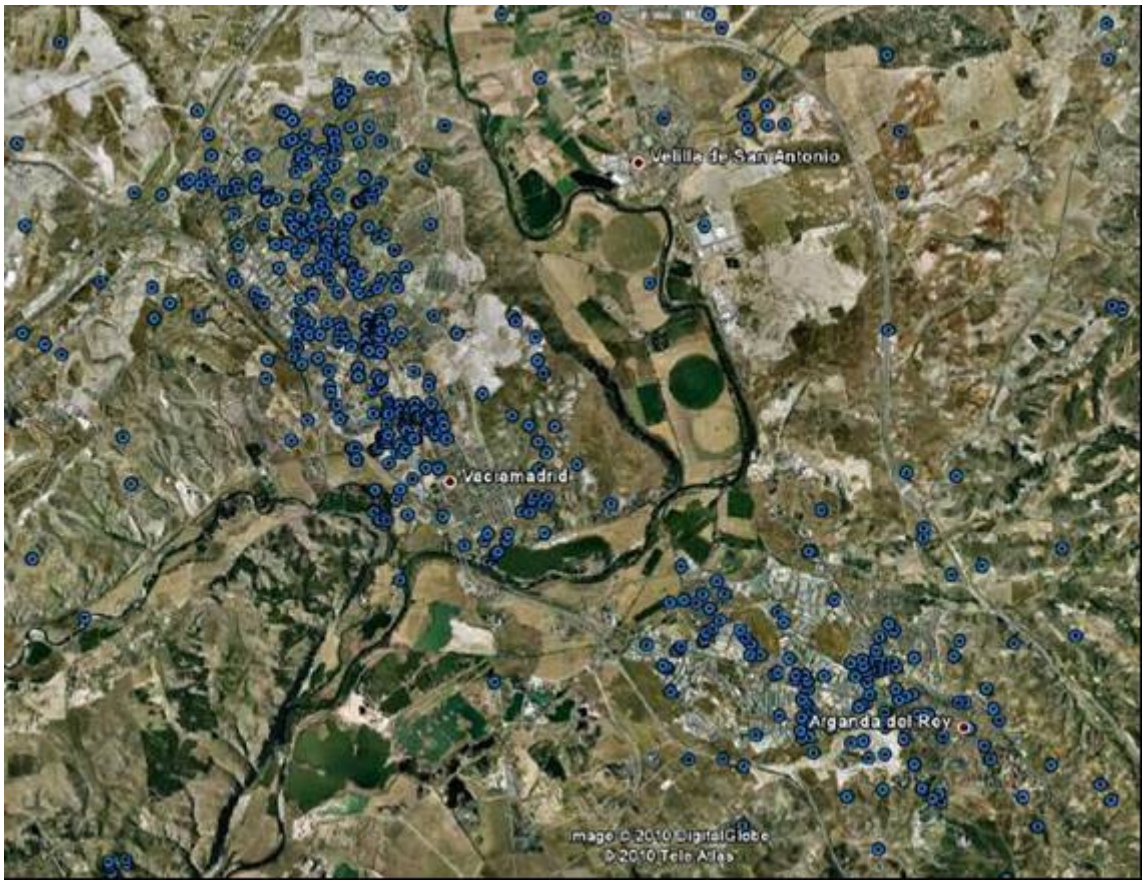

Figura 26: Ejemplo de mapa de llamadas caídas geolocalizadas

Con estos mapas temáticos el operador puede detectar de forma sencilla zonas sensibles de señal o de fallos de red para focalizar los esfuerzos de optimización en ellas.

3.5. Consideraciones de diseño

- **Requisitos**: Los requisitos para la implantación real del sistema no son muy restrictivos, dado que con un grupo de terminales de gama media-alta serviría para arrancar la monitorización de señal y del otro lado de la red, la capacidad de procesamiento dependerá del volumen real de datos recogidos, por lo que la solución es totalmente escalable. Como veremos en la fase de implementación, no es imprescindible la incorporación en los terminales de módulo GPS, dado que el sistema funciona con otros tipos de localización, pero las diferencias son tan

notables que pediremos a los usuarios la utilización de GPS en la captura de datos con el fin de arrojar resultados con la mayor calidad posible.

- **Tecnologías Consideradas:** En el diseño hemos hablado indistintamente de UMTS y GSM. En la actualidad el volumen de tráfico de voz cursado por ambas tecnologías se encuentra en torno al 50% por lo que mantiene el mismo peso en red la red GSM tradicional frente al uso de 3G. Las diferencias en volumen de tráfico se encuentran en la parte de datos pero el objetivo del Proyecto se centra en las llamadas de voz y en las medidas de cobertura, independientemente del servicio de datos que se active. Por otra parte, se ha tenido en cuenta para la recolección de muestras si el terminal se encuentra en reposo (Idle) o en conversación (dedicado). Los resultados varían en función de este estado porque el terminal mide intensidad de señal de forma diferente y traspasa la conexión de una celda a otra de manera distinta: *handover* en dedicado y reselección en Idle. Asimismo, destacar que los móviles no reportan la mejor señal en cada punto, sino la señal de la estación a la que están conectados ya que existe histéresis en el traspaso y ciertos parámetros que rigen la conexión actual en cada celda. Para conocer la señal más potente en cada punto es necesaria la utilización de scanner, equipos específicos montados en los equipos de medida o *Drive test*. Aun así, las medidas reportadas caracterizan de manera fiel le percepción real de usuario, que es el objetivo final del estudio.
- **Presentación local:** En ningún momento hemos hablado de la interfaz gráfica de la aplicación instalada en los terminales móviles. El objetivo es que dicha aplicación se ejecute en segundo plano y no sea manipulable en ningún caso por el usuario para garantizar la integridad de las muestras. La captura y transmisión de datos debe ser transparente al usuario. Únicamente se incluiría en la instalación un pequeño ítem de interacción para poder activar y desactivar la ejecución del programa, dado que el usuario es dueño de utilizar los recursos a voluntad. Esta es una de las principales diferencias con la solución implementada, dado que sí se integra una Presentación gráfica con menús para poder manipular determinadas funcionalidades como veremos más adelante.
- **Modelo de explotación:** El sistema distribuido que se nutre de la utilización personal del cliente debe integrarse en el día a día de la manera más suave posible, entre otras cosas porque se está tratando con información personal y privada de los usuarios. Es por eso que la primera solución pasa por probar la aplicación en corporaciones concretas que hagan uso del servicio de forma controlada. Por ejemplo, distribuir estos terminales en la flota de autobuses de las ciudades, en agrupaciones de taxis, en las universidades, etc. y cuya autorización de uso es gestionada por la entidad superior que facilita los terminales.

Aunque esta solución no abarca el objetivo que se plantea, que no es más que el de mejorar la experiencia del usuario y bucear en todos aquellos problemas de señal que se escapan con los métodos clásicos de optimización, y esto sólo puede lograrse extendiendo el uso de la solución a la mayoría de la cartera de clientes o al menos a una buena parte que proporcione consistencia estadística en las medidas.

Se debe tratar de compensar al cliente por los perjuicios ocasionados en la ejecución de estos programas ya que necesariamente consumirán recursos de CPU, memoria y batería, por no hablar de que la información de localización puntual del usuario debe ser autorizado por el mismo, además de cumplir con todas las normativas de protección de datos.

Por lo tanto el impacto en la factura debe ser nulo y esto no es difícil de conseguir dado que puede configurarse tarificación especial (con coste cero) para todas aquellas conexiones que se realicen contra el servidor central de la aplicación, ya sea por conexión de datos o por SMS. Incluso pueden lanzarse ofertas comerciales para tratar de captar más usuarios en la red de monitorización, tales como descuentos en factura, bonos de SMS, o puntos extra en los programas habituales de los operadores y que permiten la renovación de terminales.

4. Implementación

En esta sección se describe en detalle la solución implementada, así como las decisiones tomadas a la hora desarrollar el sistema. La práctica totalidad de las horas de desarrollo se han centrado en la aplicación cliente, construida en Android para móviles, mientras que para el procesado de datos, se han utilizado aplicaciones propietarias para PC. En los próximos capítulos se describe la manera en que se ha construido la aplicación y cómo funciona en escenarios reales.

4.1. Desarrollo en Android

4.1.1. Motivación

Una vez conocidas las necesidades del sistema, el siguiente paso fue buscar cómo llevarlas a cabo. De entre todas las posibilidades que ofrece el universo de Sistemas Operativos móviles, la primera decisión de implementación fue elegir Android.

Figura 27: Icono Android

Tres han sido los motivos:

1. Aprovechar la experiencia en programación Java adquirida durante toda la formación académica.

2. Filosofía de código abierto de la plataforma.

3. Y la enorme explosión en cuota de mercado de Android desde su aparición.

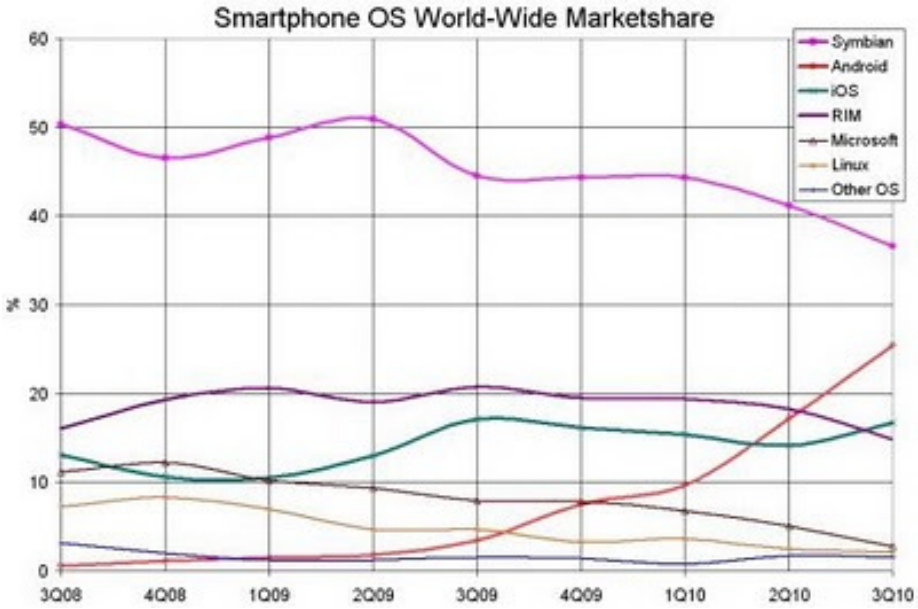

Figura 28: Evolución Cuota Mercado Mundial SSOO Móviles [8]

La elección no limita en absoluto el carácter universal del sistema. Es posible implementarlo en cualquier plataforma aunque para la elaboración del Proyecto únicamente se haya desarrollado en Android.

4.1.2. Origen de Android

Android es una plataforma de software y un sistema operativo para dispositivos móviles basada en un kernel Linux, desarrollada por Google. Esta plataforma permite a los desarrolladores escribir código en Java que se ejecute en móviles mediante las librerías Java desarrolladas por Google. La mayor parte de la plataforma de Android está disponible bajo licencia de software libre.

En julio de 2005, Google adquirió Android, Inc., una pequeña empresa de California. En esos momentos, la compañía se dedicaba a la creación de software para teléfonos móviles. Una vez en Google, el equipo desarrolló un S.O. basado en Linux para dispositivos móviles. Más adelante, Google adaptó su buscador y sus aplicaciones para este nuevo entorno.

En septiembre del 2007, Google tenía varias patentes de aplicaciones sobre el área de la telefonía móvil. El 5 de noviembre del mismo año, se anunció la fundación de la *Open Handset Alliance* ^{TM} [9] al mismo tiempo que la creación de la plataforma Android. La Open Handset Alliance está formada por un consorcio de 34 compañías de hardware, software y telecomunicaciones, dedicadas a investigar estándares abiertos para dispositivos móviles.

El primer teléfono en el mercado que posee Android es el T-Mobile G1, lanzado el día 22 de octubre de 2008 con la versión Android 1.0 preinstalada.

4.1.3. Arquitectura de Android

A continuación se representa de forma gráfica la arquitectura de la plataforma Android, en forma de "pila" en la que cada bloque se vale de los servicios que proporcionan las capas inferiores:

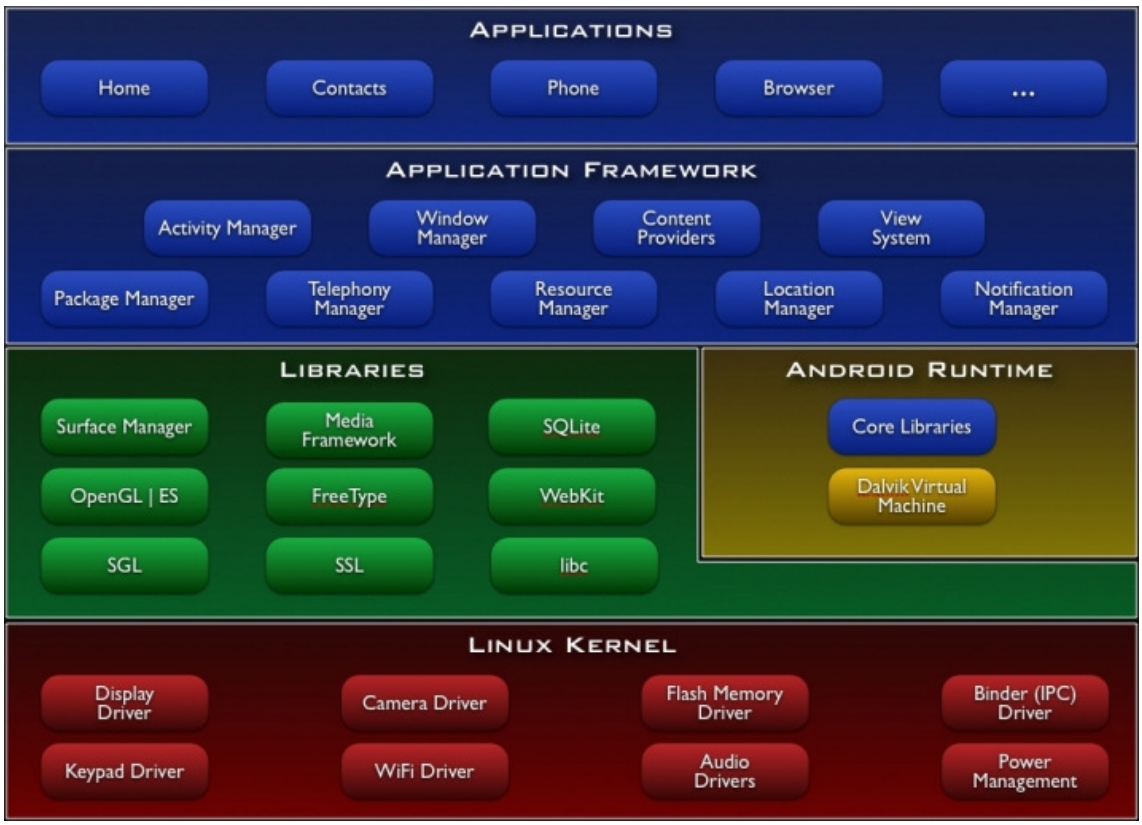

Figura 29: Arquitectura Android

- **Núcleo - Linux**: Android depende de la versión de Linux 2.6 como capa de abstracción entre el hardware y el resto de elementos de la pila. Gestiona los servicios de nivel hardware del sistema, como seguridad, gestión de memoria, gestión de procesos, pila de red, y modelo de drivers. La elección de este núcleo permite a Android ser compatible con muchos de los drivers creados para linux, facilitando en gran medida el desarrollo sobre la plataforma.
- **Runtime de Android**: Android incluye un set de librerías "core" que proveen la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. La máquina virtual ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. Ejecuta ficheros .dex empaquetados en un .apk (compresión en formato zip, similar al usado por java con los .jar) y permite traducir las .class de JAVA a .dex

- **Librerías:** En esta capa se incluye el set de librerías desarrolladas en C/C++, en las que se basan el resto de componentes del sistema. Se trata de las librerías un nivel por debajo en abstracción de las librerías Java y que aportan las funcionalidades de control gráfico, multimedia, red, etc. Se presentan hacia la capa superior a través del *framework* de aplicaciones de Android, el cual interactúa con las librerías mediante JNI (*Java Native Interface*). Algunas son: System C library (implementación librería C estándar), librerías de medios, librerías de gráficos, 3d, SQLite, entre otras.
- **Framework de aplicaciones:** Los desarrolladores tienen acceso completo a las APIs del *framework* usado por las aplicaciones base. La arquitectura está diseñada para simplificar el reuso de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades. Éste mismo mecanismo permite que los componentes sean reemplazados por el usuario.
- **Aplicaciones:** Es el nivel que se encuentra en la parte superior de la arquitectura. Contiene tanto las aplicaciones por defecto en el sistema como todas aquellas creadas por el usuario. Todas las aplicaciones están escritas en el lenguaje de programación Java.

El kit de desarrollo de software (software development kit o **SDK**) incluye un conjunto de herramientas de desarrollo, tales como un debugger, librerías, un emulador, documentación, código de ejemplo y tutoriales. Está soportado en S.O. Windows, Linux y Mac.

4.1.4. Entorno de desarrollo

El entorno de desarrollo (*integrated development environment* o IDE) comprende un conjunto de herramientas de programación que facilitan el desarrollo Software ya que aúne funcionalidades de edición, compilación, depuración, presentación y simulación de código. Son imprescindibles para la construcción de interfaz gráfica y su objetivo es proveer un marco de trabajo amigable para la mayoría de los lenguajes de programación.

Para el desarrollo de la aplicación se ha utilizado la herramienta Eclipse [10] :

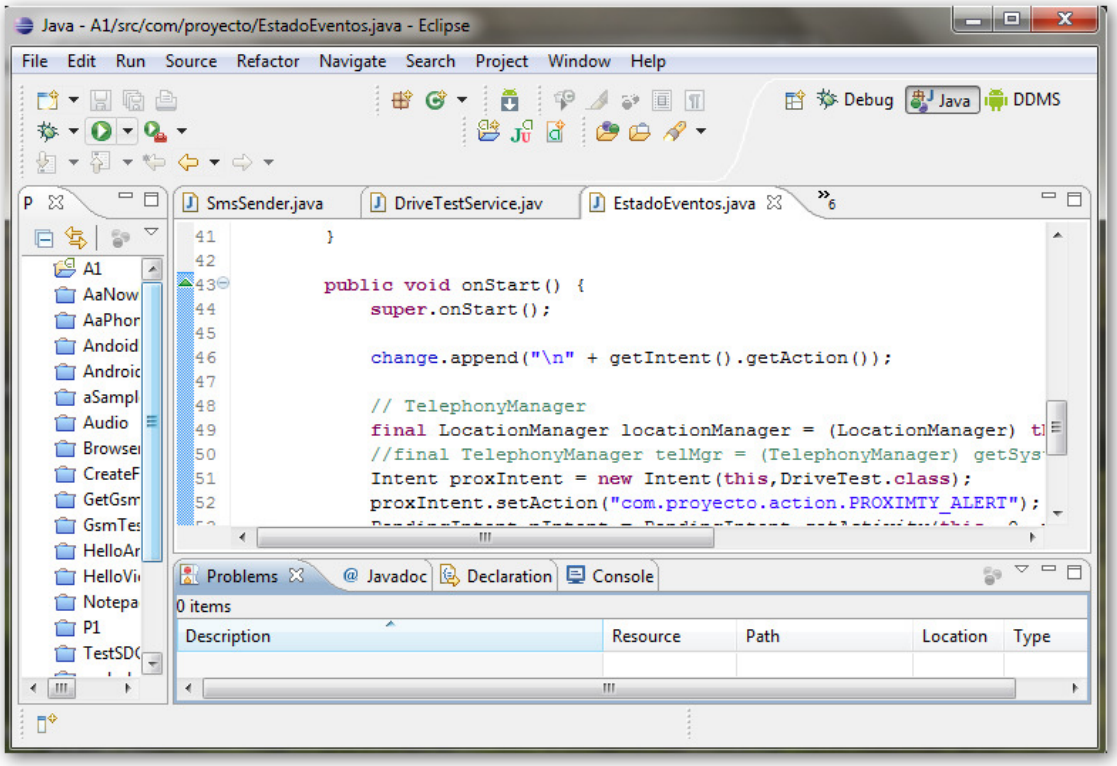

Figura 30: Vista principal de Eclipse

Eclipse permite desarrollar en casi cualquier lenguaje de programación y permite la instalación de *plugins* específicos que enriquecen la experiencia de desarrollo para determinadas plataformas. En el caso de Android, es imprescindible la instalación del plugin ADT (*Android Development Tools*) proporcionado por Google. Entre sus funcionalidades incluye la posibilidad de generar un terminal virtual para simulación que presenta fielmente el comportamiento de un terminal móvil real. Además se incluyen herramientas de depuración como DDMS (*Dalvik Debug Monitor Server*) que además de permitir de forma cómoda la detección de errores incluye el control del emulador para poder enviar mensajes o llamadas hacia el dispositivo virtual.

Eclipse es el IDE recomendado por Google para desarrollo.

4.1.5. Componentes de una aplicación Android

Cualquier aplicación en Android está compuesta por al menos uno de los 4 componentes que veremos a continuación. Todos ellos deben estar definidos en un fichero de configuración llamado Manifiesto de Android, escrito en XML (AndroidManifest.xml) y que es imprescindible para cualquier aplicación Android. Además de la definición de los bloques anteriores, el manifiesto contiene otras definiciones globales, así como permisos de ejecución. No forma parte del código pero rige la manera en que se construye la aplicación.

A continuación se exponen los cuatro tipos de componentes que conforman una aplicación para Android:

- **Activity:** Las *Activities* son el elemento Android más común. Presenta la Interfaz de Usuario con la que interactuar en la aplicación. Está formado por diferentes "*Views*" (o Vistas, que son cada uno de los elementos gráficos que componen una *Activity*) y son el equivalente a una ventana o cuadro de diálogo en una aplicación de escritorio. Las *Activitie*s no contemplan únicamente el contenido gráfico sino que contienen código cuya semántica está directamente relacionada con la interfaz que representan.

  Como veremos más adelante, una Actividad tiene un ciclo de vida concreto y se ejecuta en función de la visibilidad que tenga el usuario de ella. Así, al pasar a segundo plano su actividad queda en pausa hasta que el "foco" de la aplicación vuelve a ella. Las actividades se implementan en Android mediante clases independientes.

  La comunicación entre diferentes *Activities* se realiza mediante ***Intents***. Los *Intents* son los mensajes del sistema encargados de notificar eventos (p.e: Tarjeta SD Insertada) y pueden definirse dos tipos: *Intent* con destino único o *Broadcast Intents* (Varias aplicaciones lo "escuchan")

- **Broadcast Receiver:** Un *broadcast receiver* es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: "Batería baja", "SMS recibido") o por otras aplicaciones (cualquier aplicación puede generar mensajes)
- **Service:** Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (*activities*) si se necesita en algún momento la interacción con del usuario. Se utilizan para operaciones que requieren un periodo de ejecución largo, en comparación con las *Activities*, como la reproducción de música.
- **Content Provider:** Un *content provider* es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los *content provider* que se hayan definido.

Ciclo de vida de las actividades en Android:

Toda *Activity* sigue un ciclo, el paso entre estados puede deberse a la ejecución de código o a la intervención del usuario. Cabe destacar que, aunque no introduzcamos nada de código en algunos estados, la *Activity* sigue pasando por ellos.

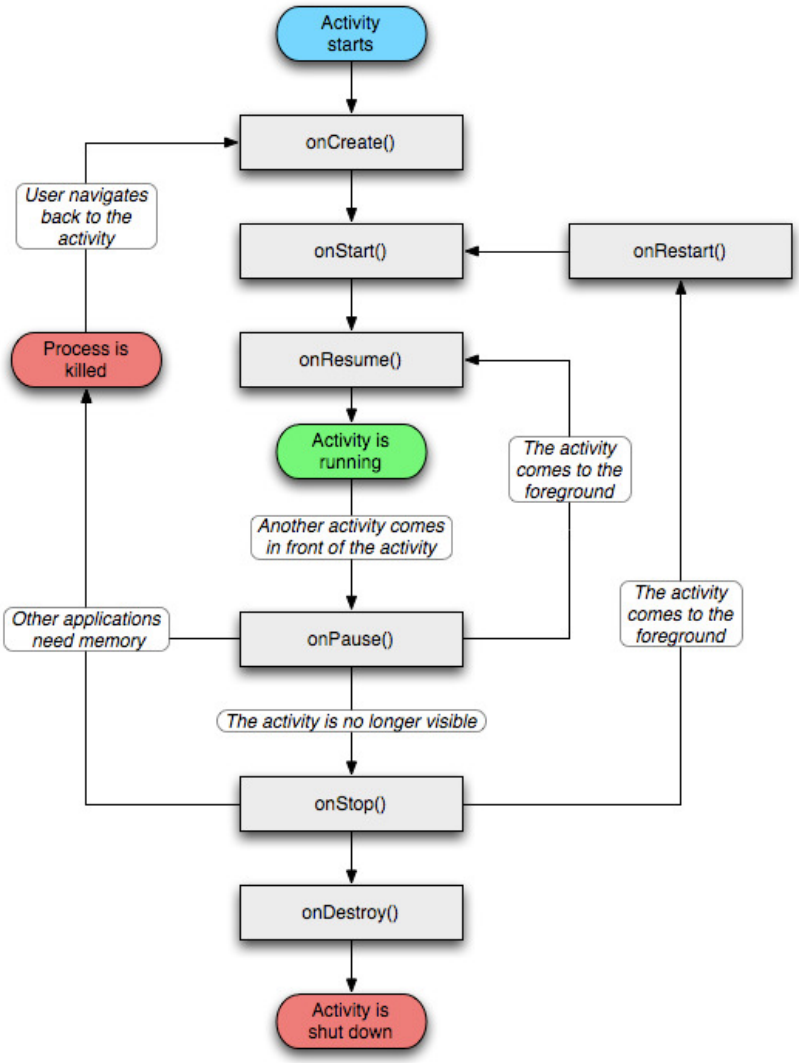

Figura 31: Ciclo de vida de un objeto *Activity*[11]

En a figura anterior puede verse la secuencia de estados por los que pasa toda Actividad en Android y en la siguiente se muestra el grado de visibilidad que tiene para el usuario cada uno de los estados:

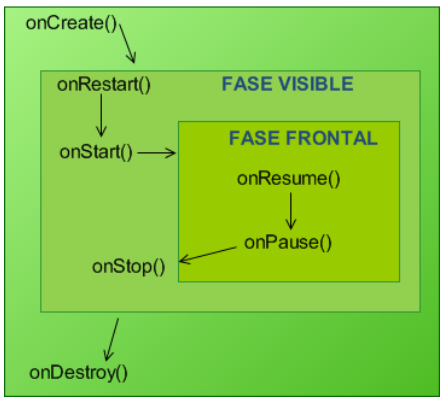

Figura 32: Visibilidad estados clase Activity

En esta sección simplemente se ha presentado de manera breve el entorno de trabajo y lenguaje de programación utilizado, de esta forma podemos contextualizar la manera en que se ha desarrollado la aplicación. En los siguientes apartados se describen las soluciones concretas de implementación, teniendo en cuenta la plataforma escogida.

4.2. Arquitectura del sistema monitorización

Antes de explicar uno a uno los elementos software de los que consta la aplicación desarrollada, retomamos el esquema de arquitectura visto en la fase de diseño, con las modificaciones que se contemplan en esta fase de implementación. Nos servirán para comprender en los siguientes apartados, las decisiones tomadas durante el desarrollo.

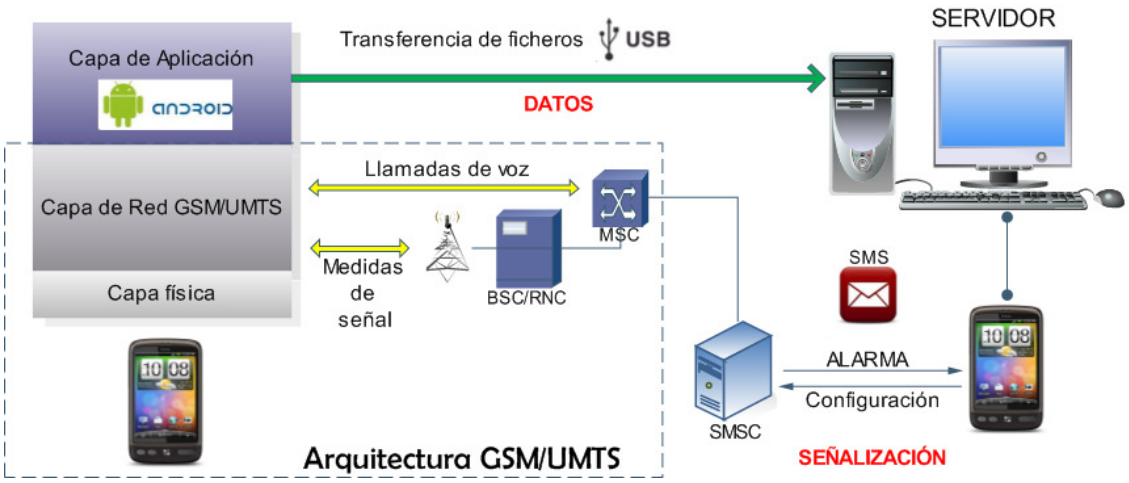

Figura 33: Arquitectura real del sistema

El comportamiento del terminal en la captura de muestras y detección de llamadas caídas es idéntico al explicado en la fase de diseño. Depende únicamente de la interacción natural del móvil con la red.

La diferencia estriba en la comunicación con el servidor. Para la implementación del sistema se han considerado dos planos de comunicación: **señalización** (vía SMS) y **datos** (vía USB). Señalización para intercambio de mensajes de configuración entre cliente/servidor y el plano de datos para la transferencia de los ficheros de recolección de muestras. Ambos tipos de comunicación quedan explicados en el apartado 4.10.

4.3. Descripción general de la aplicación

En esta sección se exponen las diferencias entre la solución implementada y la versión de diseño, describiendo de nuevo y de forma general las funcionalidades del sistema. Tomemos como hilo conductor el diagrama de flujo:

4.3.1. Diagrama de flujo

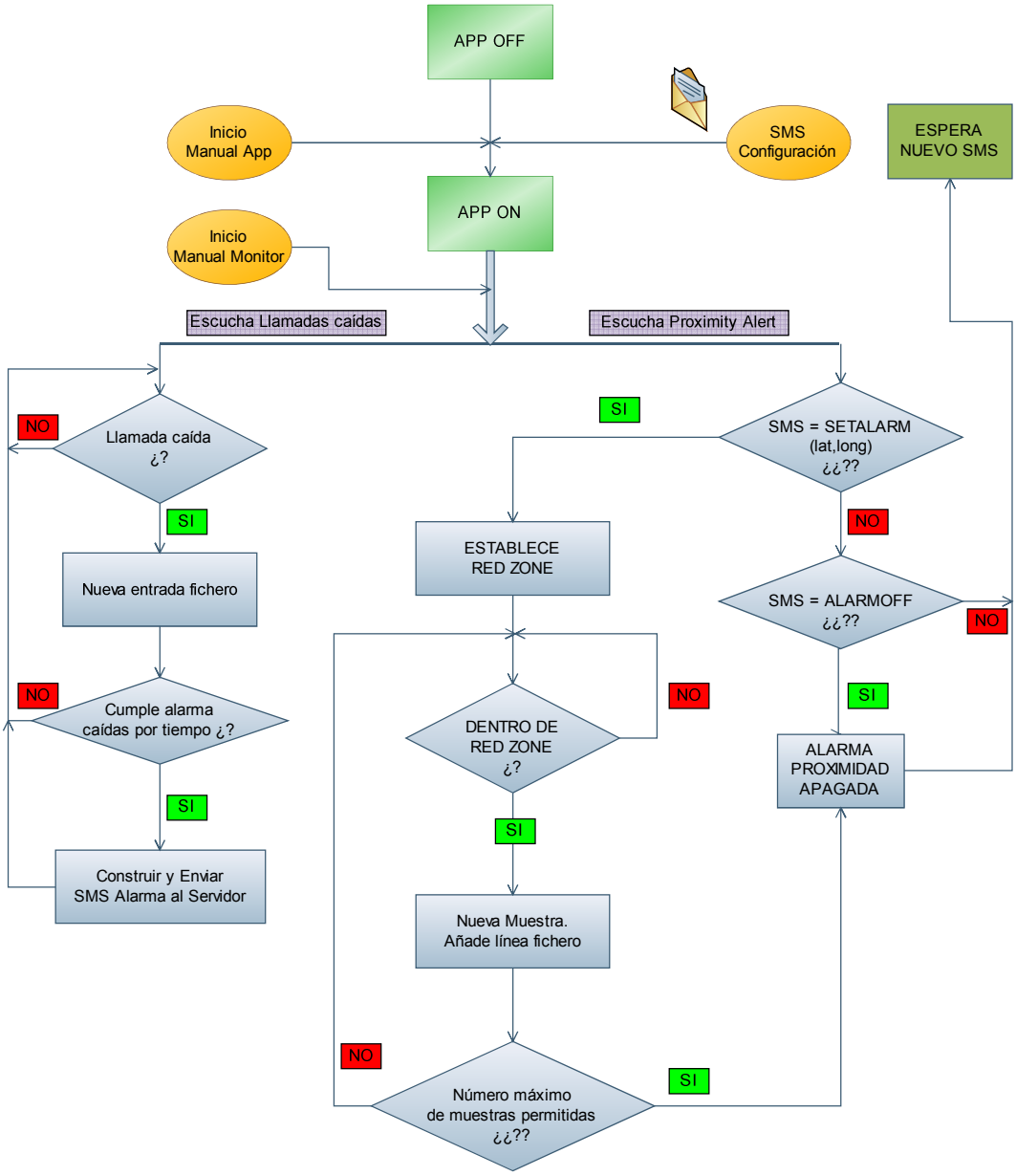

Figura 34: Diagrama de flujo de la aplicación real

La monitorización del sistema puede arrancarse de dos formas: Manual o vía SMS. La primera requiere tener la aplicación en ejecución para activar la monitorización de forma manual (simplemente un botón en la pantalla de inicio.

Consultar apartado 4.11)

Una vez que la monitorización está en ejecución el sistema reacciona a dos estímulos: llamadas caídas y detección de zona de medida. Por eso se representan dos ramas en el diagrama de flujo. En este momento la aplicación no realiza ninguna actividad si no se da ninguno de los casos anteriores.

1. Flujo de detección de llamada caída

Cada vez que se detecta una desconexión de llamada el sistema identifica si se trata de una desconexión normal o llamada caída, si se trata de este último caso, se almacena en un fichero local la posición y condiciones de la misma. A continuación busca en memoria si la anterior caída registrada ocurrió dentro de un instante T, lo que dispara una alarma inmediata hacia el servidor en forma de SMS (sin establecer sesión de datos como expusimos en la fase de diseño). Esté procedimiento se explica de forma detallada en el apartado 4.7

2. Flujo de detección de zona de medida

En esta rama sólo se entra si la inicialización se hace vía SMS. Al detectar el mensaje de texto se busca la palabra clave SETALARM que establece la zona a medir almacenando centro y radio de la circunferencia que encierra lo que hemos denominado "*Red Zone*". Sí el mensaje no incluye esos caracteres pero sí "ALARMOFF", se cancela la toma de muestras. En el caso de que no coincida con ningún mensaje de configuración, éste es ignorado. Consultar apartado 4.10 para ampliar información.

Volviendo al caso del mensaje de configuración, el sistema simplemente registra la zona a medir estableciendo un "*proximity Alert*", funcionalidad de una de las APIs de Android para la gestión de posicionamiento. En caso de dispararse esta alerta (al entrar en la *red zone*), comienza la captura de datos, que no finaliza hasta salir de la zona o alcanzar el número máximo de muestras. Todo el procedimiento se explica en detalle en 4.6 y 4.8

4.3.2. Bloques Funcionales

Todas las funcionalidades expuestas en el apartado anterior pueden encapsularse en 5 bloques con semántica propia y que resumen el comportamiento de la aplicación en un nivel de abstracción superior al del código java, que veremos en el apartado siguiente.

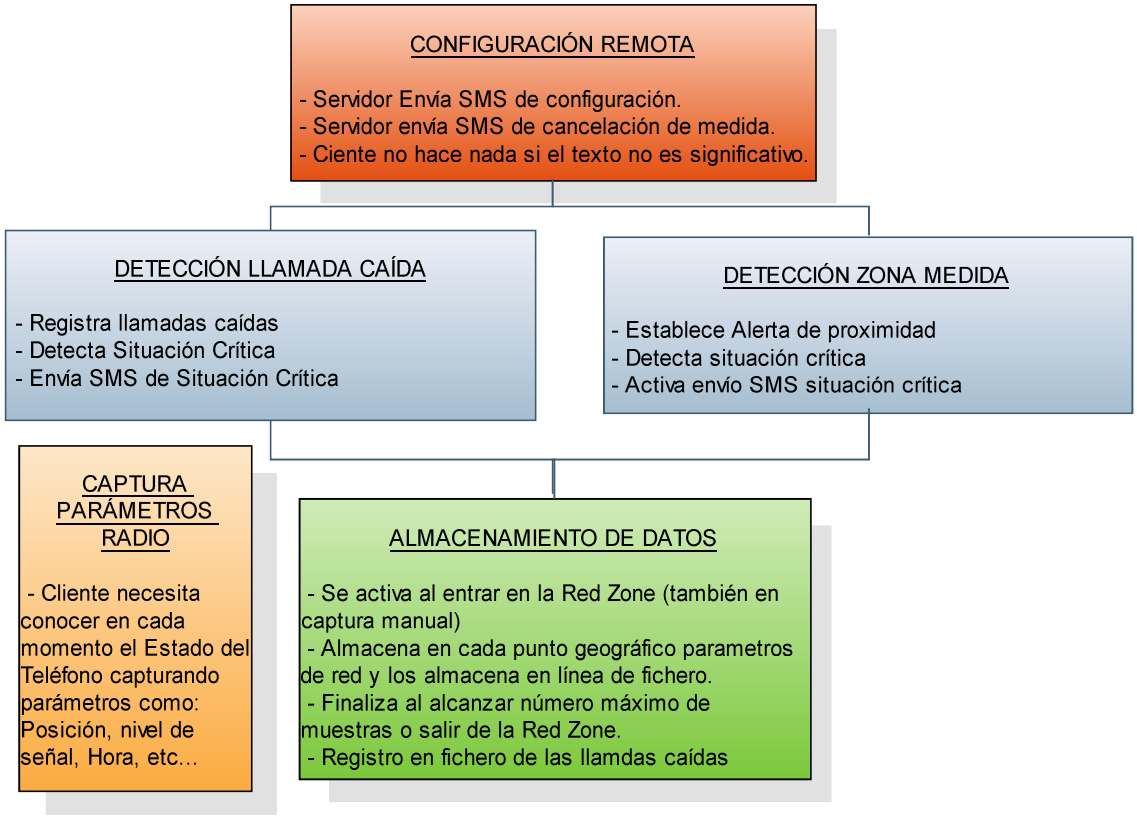

Figura 35: Diagrama de bloques del sistema

1. **Captura de Parámetros Radio:** Es el bloque encargado de actualizar en tiempo real todos los parámetros de interés desde el dispositivo (Nivel de señal, Posición, Tiempo, Operador, etc). Explicado en 4.5
2. **Configuración Remota:** Desde el servidor y vía SMS, se activa la aplicación y/o se establece la zona de medida. También se cancela la toma de muestras de forma remota. Explicado en detalle en 4.6
3. **Detección de llamada caída**: En el apartado 4.7 se expone en detalle todas las características de esta funcionalidad, que es la encargada de detectar llamadas caídas, registrarlas y actuar en consecuencia.
4. **Detección de zona de medida**: Como hemos presentado anteriormente el sistema es capaz de realizar medidas de cobertura en puntos geográficos que ordena el Servidor. El procedimiento se explica en 4.8.
5. **Almacenamiento de datos:** En el apartado 4.9 se detalla el formato del fichero resultante en la captura de eventos y cómo se ha implementado en Android.

Cada una de estas funcionalidades quedará mapeada en una o varias clases Java dentro de la aplicación Android tal y como veremos a continuación.

4.4. Esquema de Clases

Bajando un nivel más de abstracción llega el momento de describir como se han implementado cada una de las funcionalidades en el código java para la aplicación Android.

Todo el código se compone de 9 clases java repartidas en 4 clases *Activity*, 3 clases *Service*, una clase *Broadcast Receiver* y una clase neutra para almacenar constantes.

Como expusimos en la breve descripción de la plataforma Android, las actividades están formadas por Vistas que ofrecen una interfaz gráfica y los servicios ejecutan código en segundo plano sin interacción directa con el usuario. En la fase de diseño asumimos toda la ejecución de la aplicación en segundo plano y transparente al usuario por lo que sólo deberían existir clases *Services*. Pero para la implementación se han realizado partes del sistema en forma de *Activity* para presentar al usuario algunas opciones de visualización e interacción. Con la aplicación instalada el usuario va a poder arrancar la monitorización, consultar la ayuda, salir de la aplicación, etc., y estas opciones deben estar creadas a partir de *activities* en Android. En el apartado de interfaz gráfica se exponen todas las opciones de visualización (4.11)

Para explicar las decisiones de diseño en la generación de código en el siguiente gráfico peden verse las distintas clases generadas por tipos además de las relaciones entre ellos:

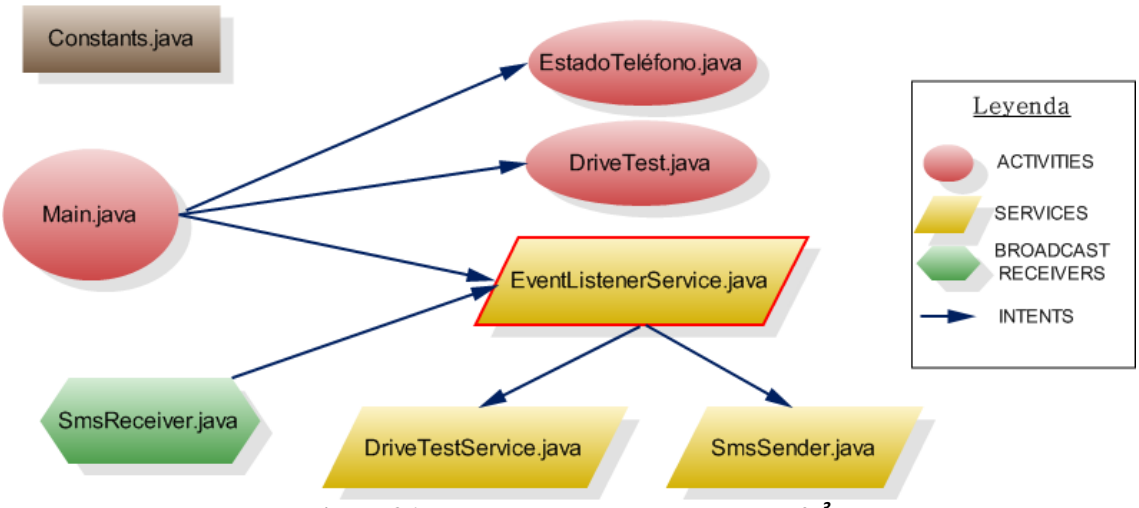

Figura 36: Esquema de clases de la *aplicación*[3]

El mecanismo por defecto de comunicación entre actividades y servicios en Android son los *Intents*. En la figura 36 se han representado mediante flechas. El conjunto de ficheros fuente de la aplicación se "empaqueta" en: `package com.proyecto`

Durante la ejecución de la aplicación, la aparición de ciertos eventos es notificada a nivel de usuario mediante *"Toasts".* Este tipo de notificaciones se presenta en forma de "*pop up*" cuyo tiempo de permanencia en pantalla es configurable así como la transparencia en su visualización:

[3] En la figura no se representa la *activity* "EstadoEventos.java". Su uso es exclusivo para depuración, mostrando una interfaz gráfica para el servicio principal "EventListenerService".

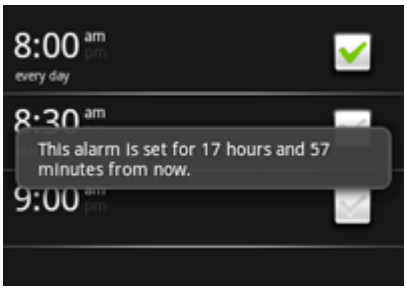

Figura 37: Ejemplo de notificación Toast

A continuación se describen las principales funcionalidades implementadas en cada clase:

4.4.1. Clase Constants

Clase compuesta únicamente por variables que comprenden constantes globales del sistema como el tiempo estipulado entre Caídas o el número máximo de muestras permitidas en la captura de datos.

4.4.2. Clase Main

Activity lanzada en la **inicialización** de la aplicación y que presenta la primera Interfaz de usuario. En el código están descritos todos los *intents* que realizan llamadas al resto de clases y *services* que interactúan con esta clase principal. Esta interacción se lleva a cabo desde la ventana principal y desde la opción de menús del terminal. El conjunto de opciones disponibles se detallan en el apartado 4.11

La *Activity Main* no posee inteligencia alguna, simplemente se encarga de presentar las opciones posibles al usuario que son activadas al pulsar en alguna de ellas. Estas son 3 de las acciones principales que se lanzan desde la ventana principal: EstadoTeléfono (4.4.3), DriveTest (4.4.4) y EventListenerService (4.4.5)

4.4.3. Clase EstadoTeléfono

Activity que únicamente puede ser iniciada desde la ventana principal al pulsar “Ver Estado Teléfono” como se muestra en 4.11. No es relevante de cara al funcionamiento del sistema pero presenta al usuario el conjunto de parámetros considerados en la aplicación en formato de tabla. Entre otros, posición, nivel de señal o tecnología usada (GSM/UMTS)

4.4.4. Clase DriveTest

Esta actividad puede lanzarse desde el menú principal mediante el botón “Captura de Muestras Manual”. Tampoco influye en el funcionamiento natural del sistema pero permite al usuario tomar muestras bajo demanda independientemente

de si se encuentra en la "red zone" o no. Al iniciar la actividad, la aplicación que comienza a tomar muestras y las almacene en fichero de texto (con formato específico como se muestra en 4.9). Al tratarse de una actividad, presenta una sencilla interfaz gráfica en la que se muestra al usuario el número de muestras tomadas y si se está midiendo (variable lógica = FALSO cuando se alcanza el número máximo de muestras). Además, en la pantalla pueden verse las muestras que se almacenan en el fichero, visualización que se refresca con cada nueva muestra.

La apariencia concreta de esta interfaz puede verse en 4.11.

4.4.5. Clase EventListenerService

Es la **clase principal** de la aplicación y se ha implementado como Service. Esto quiere decir que se ejecuta en segundo plano y únicamente presenta una notificación en forma de *Toast* al arrancar.

Como se definió desde el momento de diseño, este servicio puede arrancar de dos maneras: De forma **manual**, desde Main utilizando el botón "Iniciar Monitorización", lanzando el Toast: "ESPERANDO EVENTOS…" O de forma **remota** a través de mensaje de configuración, lanzando el Toast:"MENSAJE DE CONFIGURACIÓN".

En ambos casos arranca la "escucha" de eventos encargada de detectar las llamadas caídas. La diferencia entre ambos modos de inicialización radica en la detección de zona a medir. Cuando se inicia el código mediante mensaje de texto, en el contenido del mismo se incluye la zona geográfica a medir (4.10.1) y se inicializa la alarma de proximidad de Android (4.8), mientras que con inicialización manual únicamente se almacenan llamadas caídas. No se establece zona de medida ya que ésta puede realizarla el usuario con la opción del apartado 4.4.4 y además el objetivo es que la zona de medida se configure de forma remota.

¿Cómo diferencia la aplicación qué entidad ha iniciado el *Service*? Pues mediante los identificadores de los *Intents* que arrancan el servicio. Esta clase es inicializada mediante *Intents* lanzados desde otras dos clases: La Actividad "Main" (4.4.2) y el Broadcast Receiver "SmsReceiver" (4.4.7). Desde estas clases se etiqueta mediante *Strings* el *Intent* que puede extraerse desde el código, gracias a la sentencia:

```
String option = intent.getAction();
```

En función del contenido de *option* la clase "EventListenerService" conoce qué entidad la ha lanzado y actúa en consecuencia. De forma idéntica decodifica la etiqueta *Intent* que cancela las medidas (enviado también vía SMS 4.10.1). Los 3 *Intents* que interactúan con esta clase se etiquetan así:

- `SMS_SET_ALARM: Inicio de la clase Vía SMS`
- `LISTEN_EVENT_MANUAL: Inicio Manual`
- `SMS_SWITCHOFF_ALARM: Cancelación de las medidas vía SMS`

Por tanto, esta clase tiene **dos** misiones:

1. Detección de llamada caída: Cuyo algoritmo está definido en la **propia clase** y explicado en la sección 4.7. No sólo registra las llamadas caídas sino que posee la inteligencia de detectar zonas de riesgo cuando se dan dos llamadas interrumpidas en poco tiempo. En ese caso lanza el *service* "SmsSender" (4.4.8)

2. Configuración de alerta de proximidad, que permite al terminal detectar cuándo entra o sale de la "*red zone*" para activar o desactivar la toma de muestras. Las acciones sucesivas tras cualquiera de estos eventos no se implementan en la propia clase sino que se lanza un **nuevo *Service*** para esta tarea: "DriveTestService", explicado a continuación:

4.4.6. Clase DrivetestService

Es básicamente una copia de la *Activity* "DriveTest" (4.4.4), con la salvedad de que se implementa como servicio para ejecutarse en segundo plano y que no es lanzada de forma manual, sino que la inicia el evento "Proximity Alert" (4.8), cuyo *Intent* contiene una variable lógica que determina si se está entrando en la "red zone" o saliendo. En el primer caso comienza la toma de muestras (`medir=`**`true`**) y en el segundo se cancelan (`medir=`**`false`**).

4.4.7. Clase SmsReceiver

Es la única clase del tipo Broadcast Receiver de la aplicación. Una vez compilado el código e instalada la aplicación, el sistema Android reconoce esta clase y **registra** en el dispositivo un nuevo Broadcast Receiver que responde a determinados **estímulos** y **ejecuta** determinadas acciones, incluso con la aplicación no ejecutada. En este caso, la clase se ejecuta siempre que llega un SMS nuevo y *parsea* su contenido para detectar el formato correcto de configuración enviado por el servidor (4.10.1).

En el caso de ser un SMS convencional del usuario o no contener el formato específico, simplemente lanza un *Toast* con el texto "SMS Recibido" y traslada la responsabilidad de la gestión del SMS al sistema operativo (inclusión en la bandeja de entrada, notificación, etc.)

En caso de detectar el formato correcto de configuración, la clase "SmsReceiver" reacciona lanzando *Intents* a otras clases tal y como se expone en el apartado de configuración remota (4.6)

4.4.8. Clase SmsSender

Implementada como *Service*, esta clase es inicializada directamente por

"EventListenerService" al detectar 2 llamadas caídas en corto espacio de tiempo. Desde este servicio se construye un SMS de alarma que se envía al servidor tal y como se explica en el apartado 4.7

Tras la descripción general de cada una de las clases que componen el sistema, llega el momento de explicar con detalle las principales funcionalidades de la aplicación. En los siguientes apartados se detallan los 5 bloques semánticos presentados en 4.3.2:

- Captura de Parámetros Radio (4.5)
- Configuración Remota (4.6)
- Detección de llamada caída (4.7)
- Detección de zona de medida (4.8)
- Almacenamiento de datos (4.9)

En los apartados sucesivos se explica en detalle la comunicación Cliente/Servidor (4.10), la interfaz gráfica construida para la aplicación (4.11) y la manera en que son procesados los datos obtenidos (4.12)

4.5. Captura de Parámetros radio

Implementado en:	`EstadoTelefono.java`
	`DriveTest(Service).java`
	`EventListenerService.java`

Listeners:	`onCallStateChanged`
	`onSignalStrengthsChanged`
	`onLocationChanged`

En la aplicación en necesario conocer en todo momento y de forma actualizada ciertos parámetros (no sólo radio) para utilizar esta información en varias fases de la ejecución natural del sistema:

1. **Visualización** del Estado del teléfono por parte del usuario: (Visto en 4.4.3 y 3.1.1).

2. **Captura** de muestras de cobertura: Para cada punto se almacena la posición geográfica y los parámetros de interés para su posterior estudio.

3. **Detección** de llamada caída: Cuando ésta se produce, podemos conocer dónde ha tenido lugar, cuándo y en qué contexto.

Los parámetros capturados desde la aplicación contra el sistema operativo se realizan utilizando APIs de Android específicas. Los diferentes tipos de dato obtenidos son los expuestos en 3.1.1:

Tiempo – Posición – Sistema de posicionamiento – Nivel de señal – Estación Base – Estado de la llamada – Modo de red – IMEI – Número de Teléfono - Operador de red

Algunos pueden obtenerse con sencillas llamadas a funciones de las librerías específicas y otros parámetros son algo más complejos. Sólo se expone a continuación la obtención de los parámetros más relevantes del código:

4.5.1. Posición

Funcionalidades incluidas en el paquete Android: `android.location.*`

Para capturar la posición desde el sistema operativo en Android es necesario definir un gestor de posición `(LocationManager)` y un *listener* de cambio de posición `(LocationListener)`.

En el objeto **LocationListener** se instancia la función *onLocationChanged()* que permite definir el comportamiento de la aplicación ante cambios de posición. En la funcionalidad de "Estado del Teléfono" simplemente se presenta dicha posición al usuario.

Con las instancias a **LocationManager** obtenemos del sistema el gestor de posición: `this.getSystemService(Context.LOCATION_SERVICE)`, necesario para definir los parámetros de localización. Entre otros, la precisión que queremos garantizar en la obtención de la posición (`Criteria.ACCURACY_FINE`). Con este modificador indicamos al sistema que buscamos resolución máxima y así buscará en el dispositivo siempre por orden de mayor a menor resolución: Primero GPS, después red.

Por último, con estos dos objetos correctamente inicializados se activa la "escucha" de posición mediante **`requestLocationUpdates()`** que toma como parámetros, el criterio de resolución, el listener creado y dos parámetros muy importantes: tiempo y distancia mínimos. Si estos dos últimos están a cero, el sistema actualiza posición de manera continua, generando gran cantidad de ruido e información redundante. Por eso es posible indicarle un tiempo o una distancia neutros, en los que se ignoran cambios de posición. En nuestro sistema se consideran **10 metros** como distancia mínima entre muestras.

NOTA: En caso de no tener ningún método de posición activo (dado que en Android puede deshabilitarse) el sistema lanza un *Toast* con el texto: `"ANTES DEBES HABILITAR ALGÚN SISTEMA DE LOCALIZACIÓN EN EL TERMINAL"`. De esta manera se impide que arranque el sistema dado que sin la posición no se cumple ninguno de los objetivos propuestos.

4.5.2. Nivel de señal

Funcionalidades incluidas en el paquete Android: `android.telephony.*`

Para capturar el nivel de señal desde el sistema operativo en Android es

necesario definir un gestor de Telefonía (`TelephonyManager`) y un *listener* de cambio de estado del teléfono (`PhoneStateListener`).

Con instancias a **`TelephonyManager`** obtenemos del sistema el gestor del teléfono: `this.getSystemService(Context.`***`TELEPHONY_SERVICE`***`)`, necesario para configurar y lanzar los *listeners*. Una vez generados los objetos se define el método **`onSignalStrengthsChanged()`** que devuelve directamente desde el gestor de telefonía del dispositivo, el nivel de señal que buscamos, cada vez que éste cambia. Simplemente se le da formato, ya que varía su obtención en función de si el terminal se encuentra en GSM o UMTS.

4.5.3. Estado de la llamada

Funcionalidades incluidas en el paquete Android: `android.`**`telephony.*`**

Para capturar el estado de la llamada desde el sistema operativo en Android es necesario definir de nuevo el gestor de Telefonía (`TelephonyManager`) y un *listener* de cambio de estado del teléfono (`PhoneStateListener`).

Hasta aquí comparte código con el apartado anterior y en el *Listener* únicamente hay que definir el método **`onCallStateChanged()`** para definir el comportamiento del sistema ante cambios de estado del teléfono. El estado se almacena en la instanciación del `TelephonyManager` y puede obtenerse mediante **`getCallState()`** que devuelve un **entero** mapeado de la siguiente manera:

- 0 (Idle): Teléfono en reposo
- 1 (Ringing): Teléfono con llamada en fase “setup”
- 2 (Active): Teléfono en conversación.

Estos tres parámetros son los más complejos de obtener dado que intervienen en su captura varias clases, tanto Gestores como *Listeners*. El resto de parámetros pueden obtenerse a partir de comandos directos, como por ejemplo **`getNetworkType()`**, llamado desde la instancia del `TelephonyManager`,y que devuelve directamente la tecnología en la que acampa el terminal (GSM/UMTS).

4.6. Configuración Remota

Implementado en:	`SmsReceiver.java`
Listeners:	`onReceive`

De forma remota se activa la aplicación y se configuran sus parámetros de medida. Esta comunicación se realiza mediante SMS, cuyo formato específico se detalla en 4.10. En todas la pruebas realizadas de ha simulado el Servidor mediante un terminal móvil simplemente lanzando SMS contra el dispositivo que contiene la

aplicación. De esta manera al recibir el SMS, el sistema posee la inteligencia de detectar si se trata de mensaje de configuración o no.

Todo ello implementado en el *BroadcastReceiver* definido en la clase *SmsReceiver*, que establece un modo de actuación en el terminal incluso con la aplicación apagada. En este caso, cuando se recibe un SMS, arranca el código implementado en la clase (varias veces al día, dado que se lanza cada vez que se recibe un mensaje). Pero en pocas líneas de código (sin impacto en el consumo de recursos) parsea el texto buscando las ocurrencias que definen un mensaje de configuración.

Las funcionalidades necesarias se encuentran en el paquete Android `android.telephony.`**`SmsMessage`**, que contiene las clases necesarias para manipulación de mensajes de texto.

Todo el comportamiento se define en el método **`onReceive()`** que realiza tres tareas:

- **Diferenciar** que la notificación que lanza el sistema en forma de broadcast se refiere a SMS Recibido:

  [`if (intent.getAction().equals(SmsReceiver.`*`SMS_REC_ACTION`*`))`**]**

- **Extraer** el contenido del mensaje y trasformarlo en *String* para su manipulación.

- **Identificar** texto clave de configuración para lanzar las acciones necesarias.

Al identificar un mensaje de configuración se construye un ***Intent*** para lanzar desde esta clase el Servicio *EventListenerService* pero con un contenido especial. Los *Intents* en Android no sólo contienen una etiqueta, también pueden contener tipos de datos básicos y actuar como mensajeros entre clases Activity y/o Service. En este caso, se extrae la latitud, longitud y radio que conforman la zona de medida y se encapsula esta información en el *Intent* que lanzará la configuración de la "Red Zone". Desde el Servicio Invocado únicamente se extraen estos contenidos mediante llamadas al método: **`intent.getExtra()`**

El formato exacto de los mensajes de configuración se explica en detalle en el apartado 4.10.

Una vez que tenemos el sistema de monitorización en ejecución (se ha lanzado la clase *EventListenerService*) el sistema espera en segundo plano a que se produzcan dos eventos, llamada caída o penetración en zona de medida ("red zone"). Este es, en definitiva, el núcleo funcional de la aplicación, que se explica en detalle en los dos próximos apartados.

4.7. Detección de Llamada caída

Implementado en:	`EventListenerService.java` `SmsSender.java`

Listeners:	`onCallStateChanged`
	`onSignalStrengthsChanged`
	`onLocationChanged`

En el SDK provisto para desarrolladores en Android, se incluyen librerías de telefonía que permiten gestionar llamadas, niveles de señal, envío y recepción de SMS, etc. Sin embargo, el código fuente encargado de gestionar a bajo nivel las llamadas y el comportamiento radio del terminal (mensajes entre UE y red en los estándares de telefonía móvil) pertenece a librerías internas del núcleo de Android y no están accesibles en las APIs públicas. Esto quiere decir que en el desarrollo de aplicaciones *Third Party* en Android no pueden utilizarse directamente funciones de las APIs ocultas incluidas en `com.android.internal`. En estas librerías se incluyen funciones que identifican desconexiones de llamada en el terminal y permiten entre otras cosas, terminar una conversación colgando la llamada. Para poder utilizar estas funciones es necesario modificar el código fuente y recompilarlo, algo que desemboca en aplicaciones totalmente incompatibles entre versiones de Android y cuya ejecución puede provocar comportamientos inestables.

Por todo esto, en la implementación del sistema se tomó desde el principio la decisión de identificar las llamadas caídas desde el código y construyendo un algoritmo para su detección a partir de APIs incluidas en el SDK de Android y en concreto, en las librerías de telefonía. Se considera **llamada caída** siempre que el estado del teléfono **pase de conversación a reposo** y además el **nivel de señal esté por debajo** de un cierto umbral. Este algoritmo es más potente que la simple detección de llamada caída, dado que no sólo se identifican las llamadas con interrupción anormal sino todas aquellas que terminan de manera normal pero en malas condiciones radio, lo que desembocaría en la mayoría de los casos en fallos de red por cobertura.

Además, el registro de llamadas finalizadas tiene otro objetivo: detectar 2 llamadas caídas en un breve espacio de tiempo, lo que indicaría (fuera cual fuera el algoritmo) un problema de red que requiere el estudio en la posición geográfica donde se produce.

La manera en que se ha implementado la detección de llamadas caídas es la siguiente:

Como vimos en la obtención de parámetros radio, la aplicación conoce en cada momento el **estado de la llamada** (entero obtenido en **`onCallStateChanged()`**) y el **nivel de señal** (obtenido con **`onSignalStrengthsChanged()`**). En memoria se almacena además el estado anterior del teléfono, por lo que en el algoritmo del *Listener* **`onCallStateChanged()`** lo primero que se hace es comprobar que el estado actual es *Idle* y que el anterior era *Active*. Esto implica que ha terminado una

conversación de voz y sólo falta compararlo con el umbral de señal establecido en la clase *Constants.java*. En el caso de identificar una llamada como caída se almacenan el estado completo del terminal en un fichero de texto local donde se incluye como dato clave la **posición** en la que se produce. Para registrar todas las llamadas realizadas basta con poner el umbral en un valor alto (0 dBm) para que siempre se cumpla la condición y se registre en local el evento.

Hasta aquí la captura normal de llamada caída, pero el sistema detecta además zonas de riesgo si se producen dos llamadas seguidas. Para ellos de almacena en memoria un "timestamp" con la última desconexión producida y se compara la diferencia temporal con la nueva llamada caída p con respecto a un umbral. Este umbral también se establece en *Constants.Java* y si permite disparar ciertas acciones si se cumple la condición.

En caso de detectar dos llamadas caídas dentro de un espacio temporal concreto, arranca el algoritmo de alarma y aviso al servidor central. Este aviso se realiza vía SMS y el formato puede encontrarse en detalle en el apartado 4.10. en el código se implementa mediante el método **`sendSMS(`**`String` **`line`**`),` que toma como parámetro la cadena de caracteres que previamente ha almacenado el estado completo del teléfono.

El método `sendSms()` construye y lanza un *Intent* que contiene dicha cadena de caracteres y un identificador. El objetivo del *Intent* es "despertar" un nuevo *Service* que implementa el código de envío de SMS: *SmsSender.java*:

En esta clase se captura el texto a enviar proveniente del código anterior y se establece un número de teléfono destino (el del Servidor que se simula en El Proyecto con otro terminal). En caso de no tener el formato correcto se envía mensaje de error, pero si el destino y el mensaje presentan los formatos adecuados, se lanza desde la aplicación el SMS de alarma con `smsManager.sendTextMessage(params)`. Al igual que en la recepción de Mensajes de configuración, todas las funcionalidades de manejo de SMS en Android se obtienen de la clase `android.telephony.`**`SmsManager`**`.`

4.8. Detección Zona de medida

Implementado en:	`SmsReceiver.java`
	`EventListenerService.java`
	`DriveTestService.java`

Listeners:	`onCallStateChanged`
	`onSignalStrengthsChanged`
	`onLocationChanged`
	`onReceive`

En este apartado se desarrolla la segunda funcionalidad clave de la aplicación: La **medida de cobertura** dentro de un área geográfica concreta que establece el servidor de forma remota. El punto de partida para esta funcionalidad es la recepción

del SMS de configuración que establece en memoria el centro y radio del área a medir, como vimos en el apartado 4.6. De forma manual no se establece área de medida ya que el objetivo es configurar esta región desde el Servidor (de forma manual pueden hacerse medidas bajo demanda con sólo pulsar un botón).

Recordemos que el estado de espera de eventos se implementa en *EventListenrService* y en este código se identifica si se ha inicializado de forma manual o mediante SMS. En el segundo caso, que es el que nos interesa, la información de **latitud, longitud y radio** viene encapsulada en el *Intent* que lanzó la clase *SmsReceiver* con una etiqueta específica. Desde el código se identifica esta etiqueta y se extraen los parámetros pasados para configurar una de las funcionalidades clave dentro del código Android:

```
addProximityAlert(alertLat, alertLong, radio, -1, Intent);
```

Se lanza sobre un objeto *LocationManager* y toma como parámetro, además de la zona a medir, un *Intent* que se ejecuta cuando se **entra** o **sale** de la zona establecida.

El *ProximityAlert* es una de las funciones más interesantes de la API de localización de Android. En versiones anteriores de la aplicación se establecía como área de medida un rectángulo y se comparaba la posición actual (en latitud y longitud) con las esquinas del mismo, para determinar la pertenencia o no a la "red zone". Esto implicaba demasiadas operaciones en cada iteración, por lo que la solución adoptada finalmente fue la utilización de *ProximityAlert.* Una vez se registra la alerta en el sistema, el hilo principal de la aplicación permanece en espera hasta que se comunica desde el sistema operativo la **entrada** en la zona de medida. En este momento se lanza el *Intent* que comienza a tomar muestras de cobertura mediante *DriveTestService.java*.

La aplicación toma muestras (cada una es una línea en fichero) hasta que alcanza el número máximo establecido en *Constants.java* o sale de la zona de medida. La detección de **salida** de la "red zone" es idéntica a la de entrada (en ambos casos se lanza un *Toast* de notificación al usuario). De hecho el sistema lanza el mismo *Intent* pero lleva consigo un valor lógico que identifica si se entra o sale de la zona de medida. En caso de salir, se cancela el *flag* del código que habilita la captura de muestras.

Veamos ahora cómo se almacenan los datos capturados, ya sea por detección de llamada caída o por captura de muestras en las medidas de cobertura.

4.9. Almacenamiento de datos

Desde el principio del diseño del sistema se decidió almacenar los datos en tarjetas de memoria externas al dispositivo. Esto permite exportar los datos fácilmente a cualquier otro dispositivo y almacenar una cantidad de información mayor. Además, el soporte utilizado es *Secure Digital* (SD), que incluyen la práctica totalidad de teléfonos en la actualidad.

El formato del fichero de muestras capturado es el siguiente

Fecha	Hora	lat	long	Signal	LAC	CID	Type	Samples
20110725	8:17:36	40,325	-3,458	-81	1004	1892	GSM	1
20110725	8:17:37	40,327	-3,450	-81	1004	1892	GSM	2
20110725	8:17:38	40,329	-3,442	-81	1004	1892	GSM	3
20110725	8:17:39	40,331	-3,434	-81	1004	1892	GSM	4
20110725	8:17:40	40,333	-3,426	-81	1004	1892	GSM	5
20110725	8:17:41	40,335	-3,418	-84	1004	321	GSM	6
20110725	8:17:42	40,337	-3,410	-84	1004	321	GSM	7
20110725	8:17:43	40,339	-3,402	-84	1004	321	GSM	8
20110725	8:17:44	40,341	-3,394	-84	1004	321	GSM	9
20110725	8:17:45	40,343	-3,386	-84	1004	321	GSM	10
20110725	8:17:46	40,345	-3,378	-84	1004	321	GSM	11
20110725	8:17:47	40,347	-3,370	-82	1004	42015	GSM	12
20110725	8:17:48	40,349	-3,362	-82	1004	42015	GSM	13
20110725	8:17:49	40,351	-3,354	-82	1004	42015	GSM	14

Figura 38: formato real de fichero de captura de muestras

En él se capturan los parámetros necesarios para el posterior procesado de todos los puntos geográficos almacenados y su respectivo nivel de señal asociado. Desde el punto de vista del operador, es muy importante asociar a cada muestra el identificador de la BTS (LAC/CID,) ya que ante eventuales problemas de señal o de llamadas interrumpidas, es posible identificar la estación servidora en cada caso. En la última columna simplemente se acumula de forma ascendente el número de muestras capturadas.

El número de parámetros almacenados no contiene todos los expuestos en la fase de diseño, (se excluyen IMEI, Operador y número de teléfono) dado que se hacen las pruebas siempre en el mismo escenario. En el apartado 6.2 se explican las posibilidades de capturar estos parámetros como líneas de trabajo futuras.

En el código, una vez almacenados en memoria todos los parámetros, se construye un *String* concatenando cada uno de ellos y se ejecuta la función **`printline`**`(muestra)`. En esta función se invocan métodos clásicos de java referentes a la manipulación de ficheros y que pertenecen al paquete **`java.io.*`**. En la función se realizan capturas **try/catch** para tratamiento de errores dado que tratamos con flujos de bytes en ficheros (como `FileNotfoundException e IOException`). Si la tarjeta SD está *montada* y no hay errores se crea el fichero en la ruta:

`/sdcard/1_PROYECTO/`**`filename`**

Y si el fichero **ya existe** simplemente se permite la escritura en modo *append* (escritura a continuación). No se crea un fichero nuevo en este caso, ya que puede haber varias muestras en el mismo día y los ficheros presentan como nombre el día de la medida como veremos a continuación. Este es el formato del nombre de fichero (**`filename`**):

AAAAMMDD_**X**.txt

- **AAAAMMDD**: Identifica el Año, mes y día de la medida. Por ejemplo: 20110721_X.txt

- **X**: Carácter que identifica el modo de creación del fichero, puede tomar los valores **E, S y A**. En la aplicación se generan tres ficheros diferentes que atienden a tres situaciones distintas:

  1) Extensión E: Si se detectan llamadas caídas, se genera un fichero específico para almacenar estas muestras y se asigna el carácter E (Evento) al nombre del fichero.
  2) Extensión S: Es el caso medidas de cobertura cuando el terminal entra en la "red zone". Como se realiza en segundo plano en la ejecución de un *Service* se asigna la letra S.
  3) Extensión A: Es la toma de muestras bajo demanda por parte del usuario al pulsar el botón concreto. Como se realiza mediante una *Activity* (con interfaz de usuario) se asigna la letra A.

En el apartado siguiente se detalla el formato concreto de los mensajes que intercambian Cliente y servidor y la manera en qué esta comunicación se realiza.

4.10. Comunicación Cliente Servidor

Como se presentó en el apartado de arquitectura (4.2), la comunicación entre Terminal y Servidor Central se divide en dos planos: Señalización y Datos, ambos se detallan a continuación:

4.10.1. Comunicación para señalización

Los mensajes de señalización están formados por cadenas de caracteres de escasa longitud. En ningún caso exceden los 140 caracteres por lo que la solución adoptada para este intercambio de información ha sido la utilización de **mensajes cortos** (SMS). Además, las APIs de telefonía de Google incluyen herramientas para el fácil manejo de mensajes de texto, tanto en construcción y envío, como en recepción y procesado.

De esta manera, el terminal móvil debe ser capaz de identificar los SMS recibidos como parte del sistema de monitorización y actuar en consecuencia (ignorando los mensajes de texto del usuario). A la vez, debe ser capaz de construir también SMS de alarma cuando quiera comunicar zonas de riesgo de llamada caída al servidor. Dado que la transferencia de mensajes de señalización se realiza vía SMS, en el lado del servidor simularemos la transmisión/recepción con otro teléfono móvil.

A continuación se detallan los 3 tipos de mensajes de configuración que se

transmiten en el sistema y el formato de cada uno de ellos:

1. **Inicialización y configuración** (Downlink: Servidor → Cliente)

Desde el servidor es posible activar la aplicación, configurar el área de medida o ambas cosas a la vez. El formato del mensaje es el siguiente:

SETALARM latitud longitud radio

Ejemplo:

SETALARM 40.2503 -3.4580 0200

La palabra clave es "SETALARM" y va seguida de los valores de posición en formato decimal que representan el centro de la región de medida. Por último se añade el radio en metros, que junto con el centro, conforman una circunferencia que se establece como región de medida.

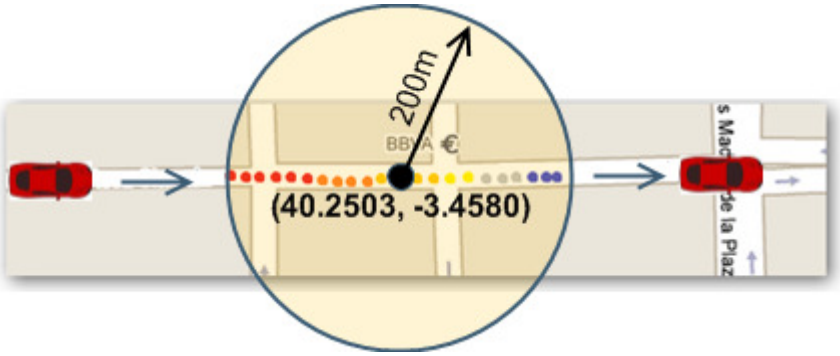

Figura 39: Ejemplo de configuración de "red zone"

Cuando el terminal recibe este mensaje, arranca la aplicación si no lo estaba y configura como zona de medida el área que encierran latitud, longitud y radio. En caso de que no se desee activar zona de medida sino únicamente activar la aplicación, es suficiente con rellenar radio = 0.

2. **Cancelación de la medida** (Downlink: Servidor → Cliente)

Con esta clave, el terminal deja de tomar muestras:

ALARMOFF

3. **Envío de alarma de zona de riesgo** (Uplink: Cliente → Servidor)

En el momento en que el terminal detecta dos llamadas seguidas interrumpidas comunica la alarma de zona de riesgo con un SMS al servidor en el que se incluye la siguiente información:

Fecha;Hora;latitud;longitud;señal;LAC;CID;tipored

Ejemplo:

20110901 ; 07:22:59 ; 40.3256 ; -3.2552 ; -99 dBm ; 1004 ; 1879 ; UMTS

Coincide con la última línea del fichero de eventos almacenada en local. No sólo se comunica qué algo extraordinario ha sucedido sino que se aporta toda la información de la que dispone la aplicación en el momento de la llamada caída: Momento exacto, posición, Celda servidora (LAC/CID) y tipo de red (GSM/UMTS)

4.10.2. Transferencia de datos

Dado que la inteligencia del sistema se concentra en la aplicación cliente, no se ha considerado como objetivo del Proyecto construir un servidor remoto para transferir los datos por la interfaz aire (vía conexión de datos GPRS/HSDPA). Una vez capturados los datos y almacenados en el teléfono se transfieren al servidor, que en la práctica es un PC doméstico, vía USB.

4.11. Interfaz gráfica y menús.

A pesar de que la ejecución de la aplicación debe ser totalmente transparente al usuario y en segundo plano, se ha creado una interfaz gráfica para interactuar con la misma y poder obtener así cierta información del comportamiento del sistema.

En la siguiente figura se muestra la ventana principal y los distintos menús:

Figura 40: Captura Interfaz principal (izquierda) y menús (derecha)

1. El Botón **Ver Estado del Teléfono** presenta la siguiente interfaz:

Figura 41: Interfaz que presenta el estado del Teléfono

Que permite conocer los valores de los parámetros de estudio en tiempo real y de un sólo vistazo.

2. **Iniciar Monitorización** lanza el *Toast* "Esperando Eventos…" y es la forma manual de inicializar la espera de eventos de llamada caída y de detección de "red zone" (la funcionalidad principal del sistema)
3. **Captura Muestras Manual** permite iniciar bajo demanda la medida de cobertura hasta que se alcance el número máximo de muestras (lanza *la Activity DriveTest*). Se abre una nueva ventana en la que se indica el número de muestras acumuladas.

Desde el botón **menú** del dispositivo se abren 4 nuevas opciones:

4. **Proximity Alert Visible:** Utilizado para depuración principalmente, presenta una ventana que espera a que el dispositivo entre en la zona de medida presentando las diferentes muestras tomadas y lanzando también la *Activity DriveTest*.
5. **Captura Muestras en 2º plano:** Idéntico comportamiento al del punto 3 pero sin interfaz gráfica. Lanza el mismo servicio (*DriveTestService*) que lanza el sistema al entrar en zona de medida pero de forma manual.
6. **Ir a ayuda Web:** Como apoyo a toda la documentación **se ha creado una página Web** [12] que sirve como manual y ayuda a la aplicación. Al pulsar el botón, el navegador integrado del dispositivo Android se abre en dicha página.
7. **Terminar y Salir:** Cancela cualquier actividad y cierra la aplicación.

4.12. Procesado de las medidas con Mapinfo Professional

En la fase de diseño comentamos que el procesado de las medidas recogidas se realiza mediante herramientas GIS, y en concreto, en todas las pruebas realizadas se han utilizado diferentes versiones *Trial* de la herramienta Mapinfo® Professional [13].

Mapinfo es una herramienta, entre otras cosas, de manipulación de mapas mediante capas superpuestas. Así, podemos presentar de forma visual capas con el perfil del terreno, carreteras, edificios, parkings, líneas ferroviarias y también la ubicación de las estaciones base.

Sobre estas capas base queremos presentar las medidas que ha generado nuestra aplicación que, como vimos en el formato de fichero, no es más que un conjunto de líneas ASCII que se ordenan en forma de tabla. Es la mejor manera de almacenar datos para Mapinfo puesto que todos los mapas que presenta la herramienta son en realidad tablas con información cartográfica. Los puntos en Mapinfo pueden contener cualquier información, por ejemplo si se trata de la capa de

estaciones de Metro, puede contener el nombre de la estación y horarios. Pero debe contener dos valores obligatoriamente, **centroideX y centroideY**, que no son más que las coordenadas del punto. El resto de tipo de figuras en Mapinfo, líneas y polígonos, se forman a partir de las anteriores.

Por tanto, en nuestros ficheros disponemos de la información geográfica de las muestras y sólo falta indicarle a la herramienta qué campo es el que corresponde a latitud y cuál a longitud. De esta manera se genera la capa oportuna y podremos manipular nuestra capa de eventos y muestras a conveniencia.

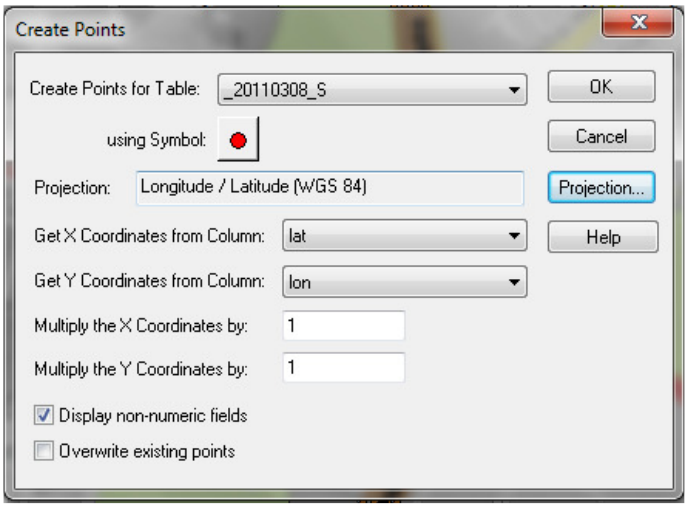

Figura 42: Generación de capa a partir de una tabla con latitud y longitud

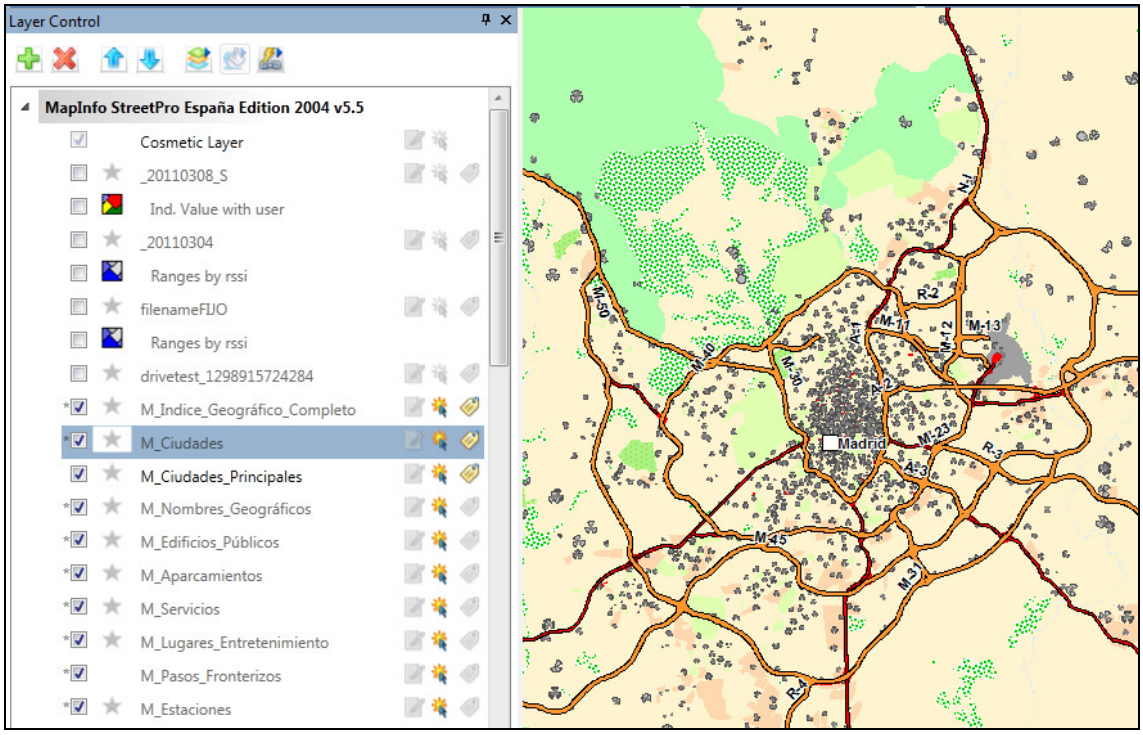

Figura 43: Estructura en forma de capas en Mapinfo

Una vez generada las capas, podremos presentar el conjunto de llamadas caídas

detectadas para tratar de detectar constelaciones de puntos especialmente críticos por la densidad en que aparecen. Y en cuanto a las medidas de cobertura que también hemos capturado en los ficheros de la aplicación, utilizamos otra de las funcionalidades que aporta la herramienta Mapinfo: las capas temáticas. Del conjunto de muestras almacenadas, pueden crearse grupos o clases por rangos de nivel de señal:

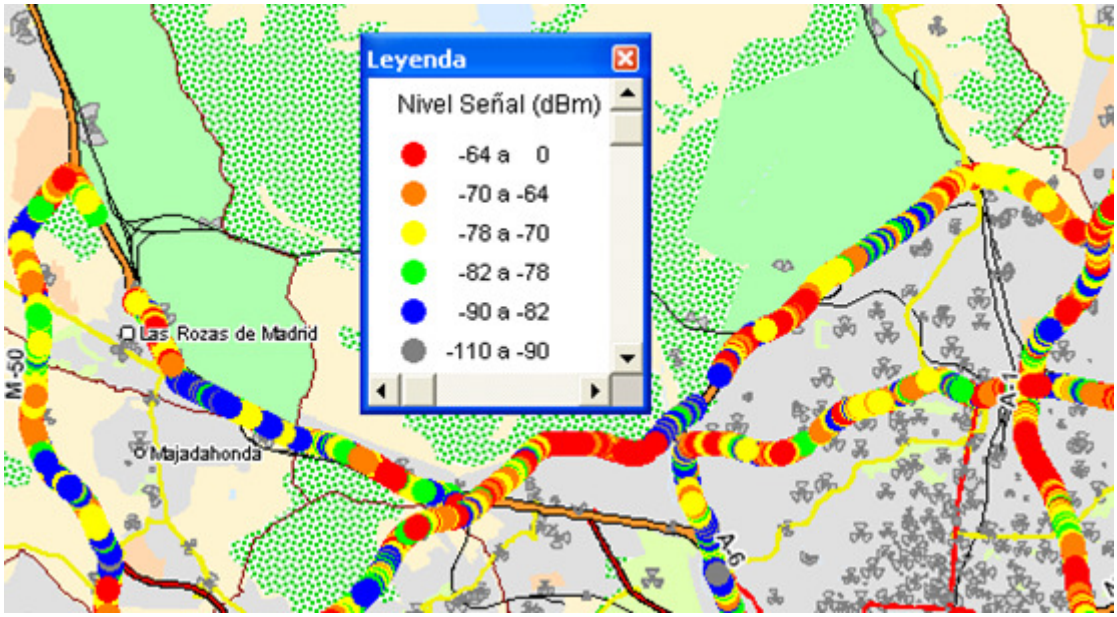

Figura 44: Ejemplo mapa temático para niveles e señal

Así, de forma visual se pueden detectar zonas con escaso nivel de cobertura para su posterior análisis. Ya sea por falta de despliegue o por avería en alguna de las estaciones que dan cobertura a dicha zona.

Ahora que se conocen en detalle las principales características del sistema, llega la hora de presentar los resultados de todas las pruebas realizadas.

5. Validación

En esta sección se presenta la manera en qué se han realizado las pruebas y cómo se han procesado los datos para la presentación de resultados. Comencemos por enumerar el material utilizado:

- **Terminal** Sony Ericsson X10 Mini, versión Android 1.6 (con módulo GPS).
- **Terminal** Orange Boston, versión Android 1.6 (con módulo GPS).
- **Ordenador Portátil** (Asus Notebook K52F) para procesado de resultados.
- **Vehículo** necesario para las medidas de larga distancia.

Con estos elementos se han recogido todos los datos necesarios para el estudio de cobertura y eventos del sistema, como se muestra a continuación.

5.1. Pruebas

Con los dispositivos móviles se han simulado todos los mensajes de configuración. Es decir, que para la parte de señalización, tanto configuración remota, como alarma de zona de riesgo, el envío y recepción de SMS se realiza con terminales móviles, en muchos casos el mismo terminal donde se ejecuta la aplicación hace las funciones de Cliente y Servidor, el PC es necesario para la parte de procesado de datos.

Por tanto, separamos en dos entidades las funcionalidades del servidor como ya expusimos en la sección de arquitectura: Para señalización SMS's directamente desde el móvil y transferencia de datos y procesado, el ordenador portátil. La mayoría de las pruebas se han realizado **en coche** para tomar mayor número de muestras y abarcar una región más amplia.

En los primeros pasos del desarrollo, el principal banco de pruebas ha sido el **emulador** incorporado en el plugin de Android para Eclipse (ADT). El aspecto del emulador es el siguiente:

Figura 45: Emulador de Android para Eclipse (ADT Plugin)

Incorpora la herramienta DDMS (Dalvik Debug Monitor Server) desde la que puede simularse comunicación con el dispositivo virtual, ya sean llamadas, SMS's o posición GPS.

Desde el emulador hemos podido probar la comunicación vía SMS entre Servidor y terminal y simular la inclusión en zona de medida (*ProximityAlert*) introduciendo manualmente latitud y longitud. Pero el emulador presentaba varios problemas que han limitado su uso exclusivamente a depuración. No es posible capturar parámetros radio relativo a niveles de señal, Identificador de celda, y el manejo de ficheros en la tarjeta SD virtual es realmente complejo. Además, para simular movimiento geográfico los valores deben ser introducidos manualmente uno a uno, haciendo tedioso el proceso de validación de funcionalidades de posicionamiento.

Por todo ello, el grueso de pruebas realizadas se ha realizado con terminales reales (lo que implicaba desplazamiento y en muchas ocasiones realizar las medidas con el ordenador portátil acompañando las mismas para posibles retoques en el código). A continuación listamos el conjunto de pruebas realizadas ordenadas cronológicamente y alineadas a los progresos en el desarrollo de la aplicación.

1. **Medida manual de cobertura:**

Esta prueba es la que confirma el correcto funcionamiento de la captura de muestras en fichero, identificando en cada punto geográfico el estado del terminal en términos de parámetros radio. El primer paso fue realizar medidas sin ningún método de localización activado en el terminal. Esto provocaba el error inmediato del sistema por lo que se definió un mecanismo de aviso al usuario para que active uno de los procedimientos de localización que incluye el teléfono. En primer lugar se realizaron varias pasadas por carreteras de Madrid con GPS y sin GPS con un límite de 500 muestras. Los resultados pueden verse en los apartados 5.2.1 y 5.2.2 donde la

resolución de uno y otro método es notable, adoptando para el resto de pruebas únicamente la localización por GPS. Es con este sistema cuando la aplicación aprovecha todo el potencial ya que se identifican los puntos con precisión de escasos metros. En el apartado 5.2.3, aunque fuera del objetivo del Proyecto, se presentan además algunos estudios de cobertura alternativos, basados en tecnología, área de BSC/RNC y por identificador de celda.

2. Detección de zona de medida:

Todas las pruebas de detección de zona de medida ("red zone") se han realizado en coche. Los diferentes resultados pueden apreciarse en la sección 5.2.4 donde se han establecido varias zonas de medida probando con diferentes radios para comprobar que el terminal **comienza a medir al entrar** y **detiene** las capturas **al salir** de la región de estudio. El procedimiento ha sido siempre el mismo: Esperar a que el teléfono haya sincronizado correctamente la señal GPS y posteriormente enviar el SMS de establecimiento de "red zone" oportuno, con el formato visto con anterioridad:

SETALARM latitud longitud radio

Esto ha permitido tomar muestras en regiones concretas y comprobar como el dispositivo no realizaba capturas en modo "free run" como en el apartado anterior, sino que es capaz de restringir las medidas a un área, cumpliendo así uno de los objetivos definidos.

3. Captura de llamadas caídas:

Para las capturas de llamadas caídas es necesario únicamente iniciar la monitorización de la aplicación, no requiere de más inputs de configuración. Cada vez que se recibe un SMS para establecimiento de zona de medida también queda el sistema listo para "escuchar" las llamadas interrumpidas además de la posibilidad manual. De estas dos formas queda el terminal preparado para la detección de estos eventos. La interacción por parte del usuario es nula, como era de esperar. Realizando un uso normal del dispositivo la aplicación en segundo plano es capaz de almacenar en disco todas aquellas situaciones consideradas como llamada caída mientras el usuario realiza llamadas que pueden o no acabar en fallo. En la sección 5.2.5 se representan todas estas capturas teniendo en cuenta diferentes umbrales de señal para la detección de caída.

4. Detección de zona de riesgo:

Continuando con el procedimiento de detección del apartado anterior se han realizado varias pruebas para la detección de zona de riesgo al capturar 2 llamadas caídas en un intervalo dado. Este intervalo se define en *Constants.java* y en todas las pruebas se ha establecido dicho umbral en **45 sg**.

Se han tomado 45 segundos como valor suficiente para que un usuario y su interlocutor, ante una llamada caída, reestablezcan la comunicación y ésta vuelva a

fallar por segunda vez. Esto implicaría un fallo en el sistema que requiere de estudio particular y por tanto se avisa de forma inmediata al servidor construyendo y enviando un SMS para tal fin. En la sección 5.2.6 se presenta un ejemplo de aviso.

5.2. Resultados

En esta sección se exponen los resultados de cada una de las pruebas presentadas en el apartado anterior:

5.2.1. Medida de cobertura con GPS

Aunque se realizaron varias pruebas de medida de cobertura, la que se expone en esta sección y en la siguiente, son representativas del modo en que las muestras son capturadas y procesadas. En este caso se realiza una medida desde sur de Madrid por la carretera M-30, hasta la calle Arturo Soria con límite 500 muestras. Iniciada manualmente:

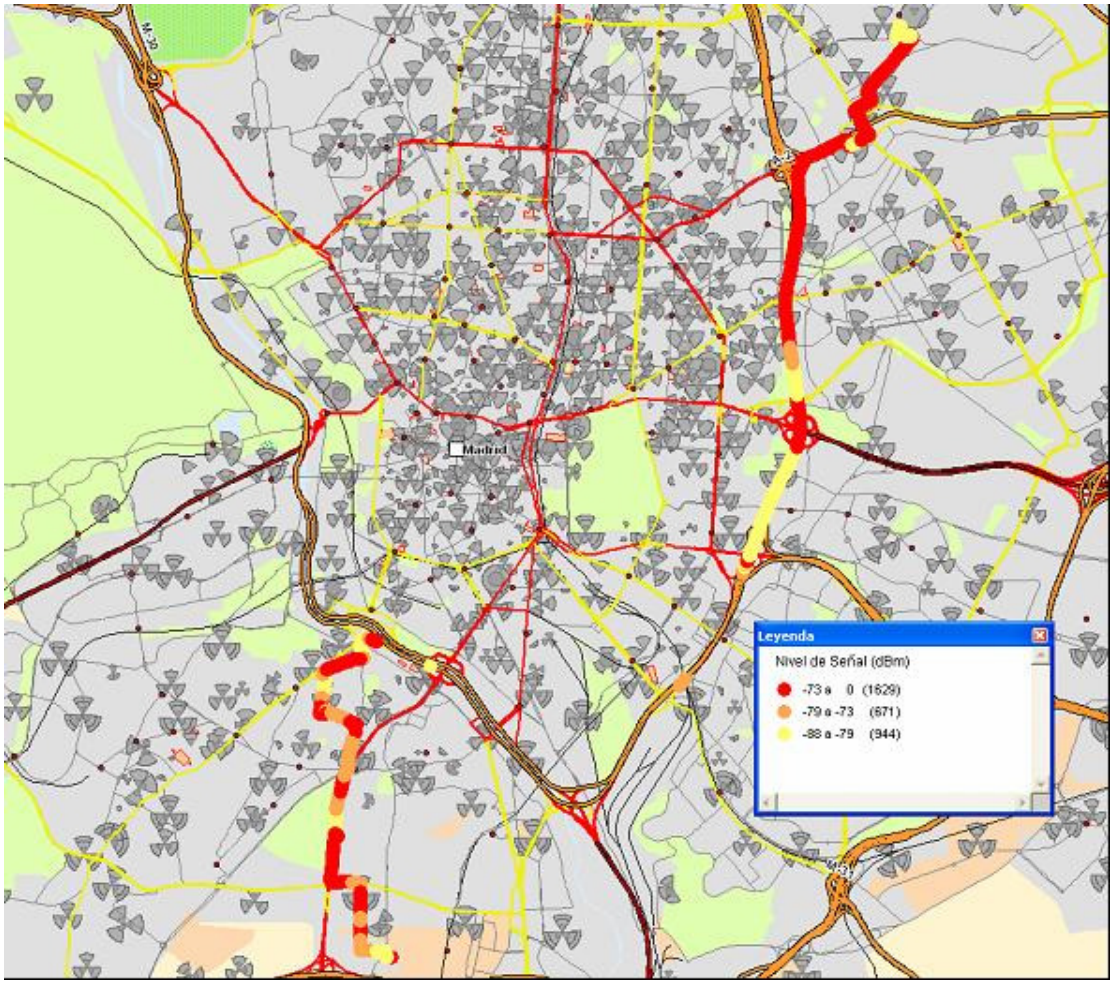

Figura 46: Medida manual. GPS Activado

Puede apreciarse la ausencia de muestras durante un tramo en el que se circula por túneles (pérdida de señal GPS). Con esta metodología, la resolución da cada muestra es lo suficientemente precisa para establecer el punto exacto donde se producen los eventos. Más aun cuando se circula por carretera.

Para presentar estas medidas de señal, como expusimos en el apartado 4.12, se trasladan los ficheros recogidos al PC donde finalmente se procesan. En definitiva el sistema arroja un fichero de texto con dos campos clave, latitud y longitud, que permite crear una capa en la herramienta Mapinfo para su posterior formato y presentación. En el gráfico de arriba se crean varios rangos de señal ene los que los más cálidos (rojos) identifican las zonas de mejor cobertura.

5.2.2. Medida de cobertura sin GPS

En esta prueba se realiza el camino inverso al de la sección anterior pero sin GPS. Como puede verse en la siguiente captura la densidad de puntos es mucho menor ya que el sistema reacciona ante cambios de posición, y en la localización **basada en red**, esta actualización es mucho menos frecuente. De hecho todo el recorrido se resume en unas 30 muestras:

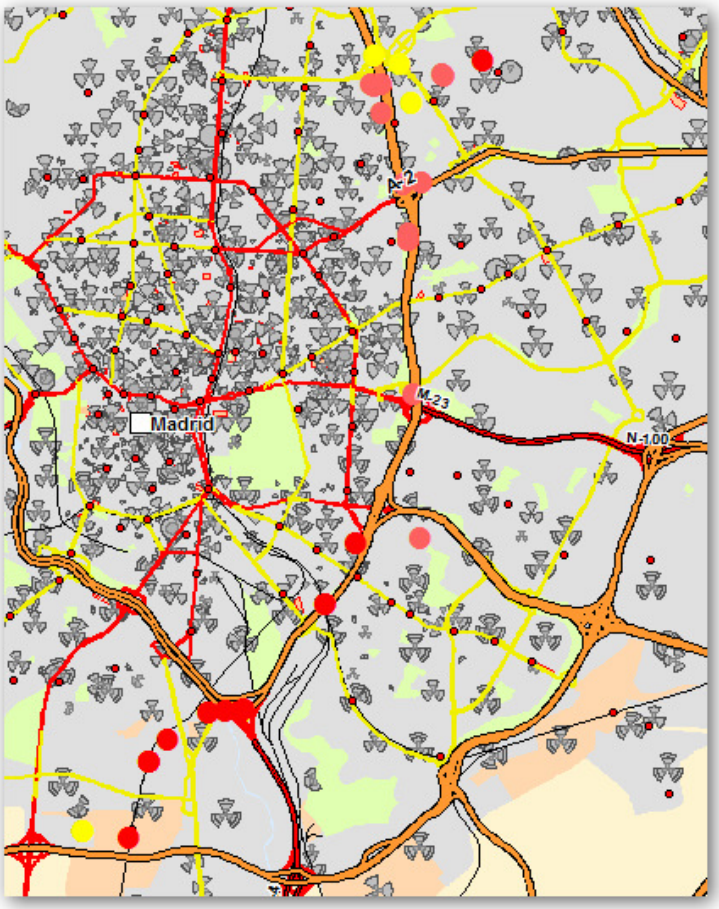

Figura 47: Medida manual. GPS Desactivado

Se representan con trazo más grueso para que sean fácilmente identificables las muestras siendo tan escasas. Este método de posicionamiento cumpliría los objetivos del Proyecto pero la precisión de las muestras es mucho menor.

5.2.3. Otros estudios de cobertura

Aprovechando el conjunto de muestras almacenado, no sólo pueden obtenerse resultados de nivel de señal, sino que asociados a otros parámetros radio, pueden caracterizarse las muestras por tecnología o por estación base como veremos a continuación.

- **Estudios** de cobertura por **tecnología**:

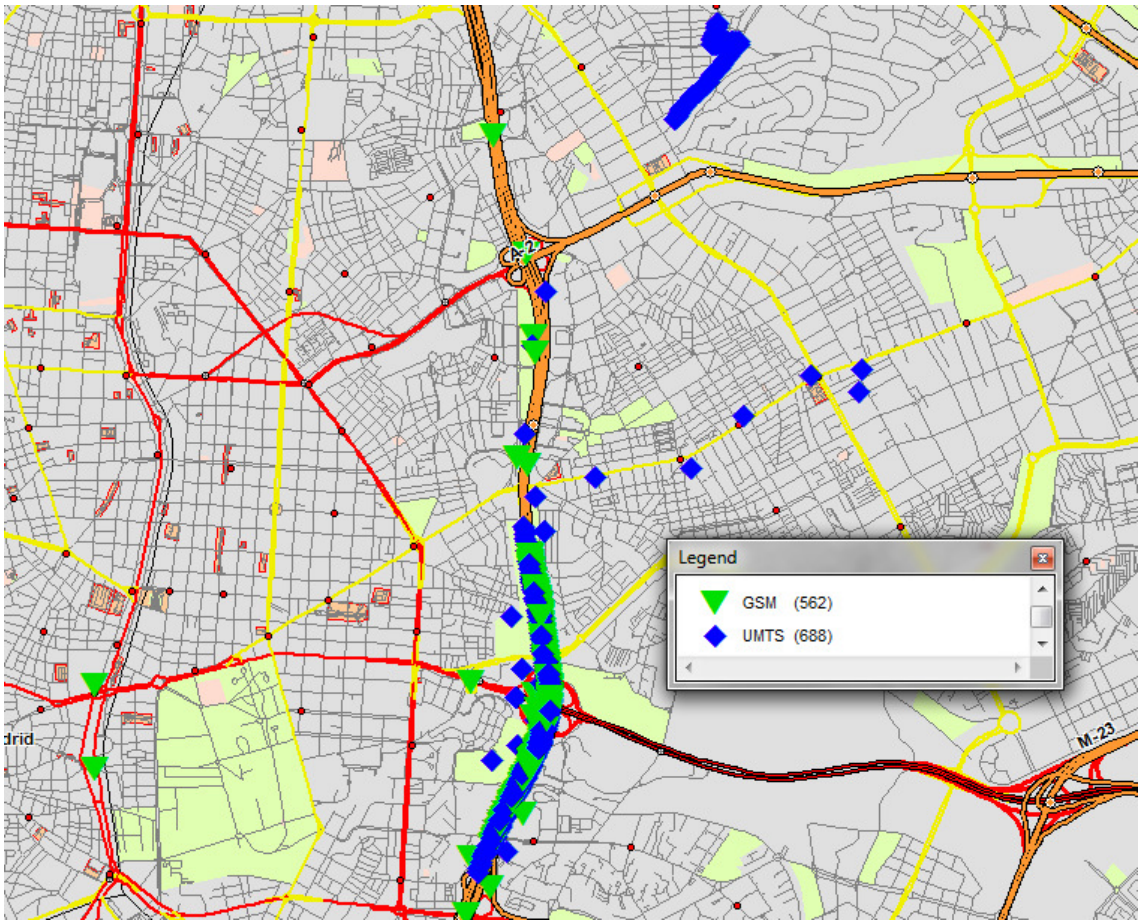

Figura 48: Comparativa GSM/UMTS

En la figura 48, simplemente se representa del total de muestras, cuántas se han capturado en GSM y cuántas en UMTS. Durante las pruebas, el terminal se encontraba en modo dual y aunque lo normal es que sea mayor el número de muestras en GSM (mejor cobertura), en el escenario presentado (en exteriores y en municipio Madrid) el número de muestras UMTS es mayor: 688 frente a 562.

En la figura 49, se representan los mismos resultados pero en forma de diagrama de barras y eliminando la componente cartográfica. Es otra manera de

realizar un estudio comparativo de niveles de señal entre tecnologías, y apodemos ver como en el extremo izquierdo gran cantidad de muestras arrojan valores muy buenos de señal para GSM. Esto es así porque gran parte del tiempo el terminal acampa en un interior donde se encuentra un despliegue especial de GSM, con antenas de interiores colocadas en el falso techo del edificio.

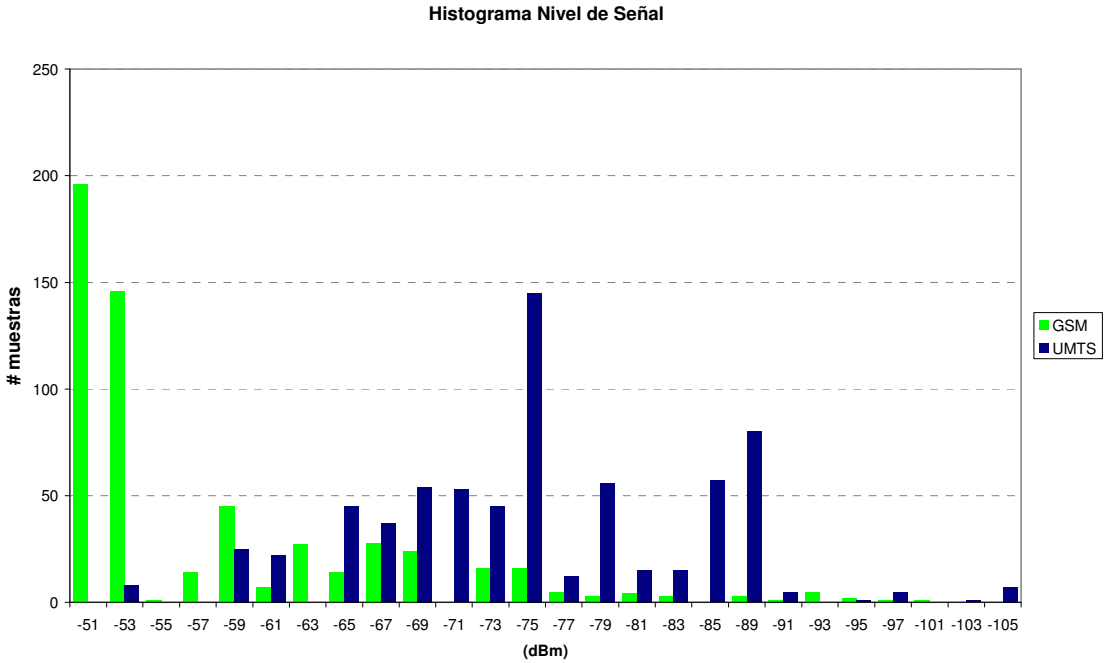

Figura 49: Histograma de muestras por tecnología

- **Estudios** de muestras por **LAC** (Local Area Code)

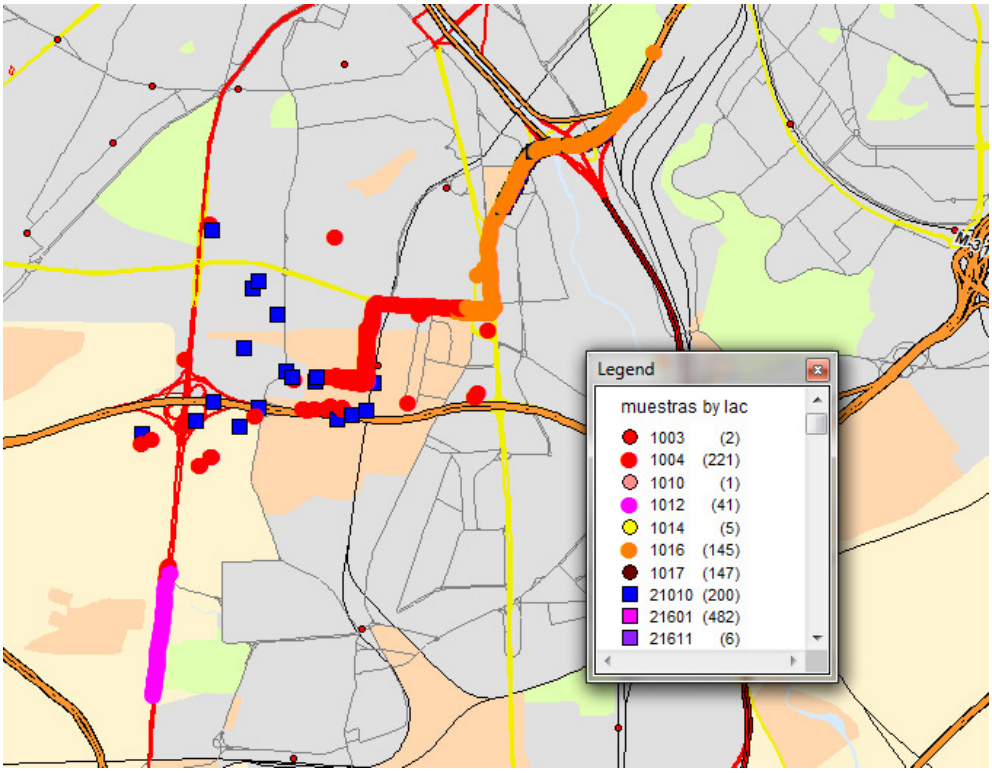

Figura 50: Estudio de muestras por LAC (Local Area Code)

En la figura 50, se presentan las muestras agrupadas por colores (en una capa temática) en función del LAC de la celda en la que acampan. El LAC es el identificador común de un conjunto de estaciones, normalmente el mismo para todas las celdas que dependen de la misma BSC/RNC, aunque también puede separarse por provincia u otro criterio de diseño. En la gráfica anterior (representado con círculos LAC's GSM y con cuadrados LAC's UMTS), se distingue claramente en qué momento de atraviesa un LAC, información útil porque en esas fronteras se incrementa la señalización y puede ser una fuente interesante de información para el Operador.

- **Estudios** de muestras por **CID** (Cell Id)

En la siguiente imagen (figura 51), se realiza el mismo estudio que en el apartado anterior pero con resolución de celda:

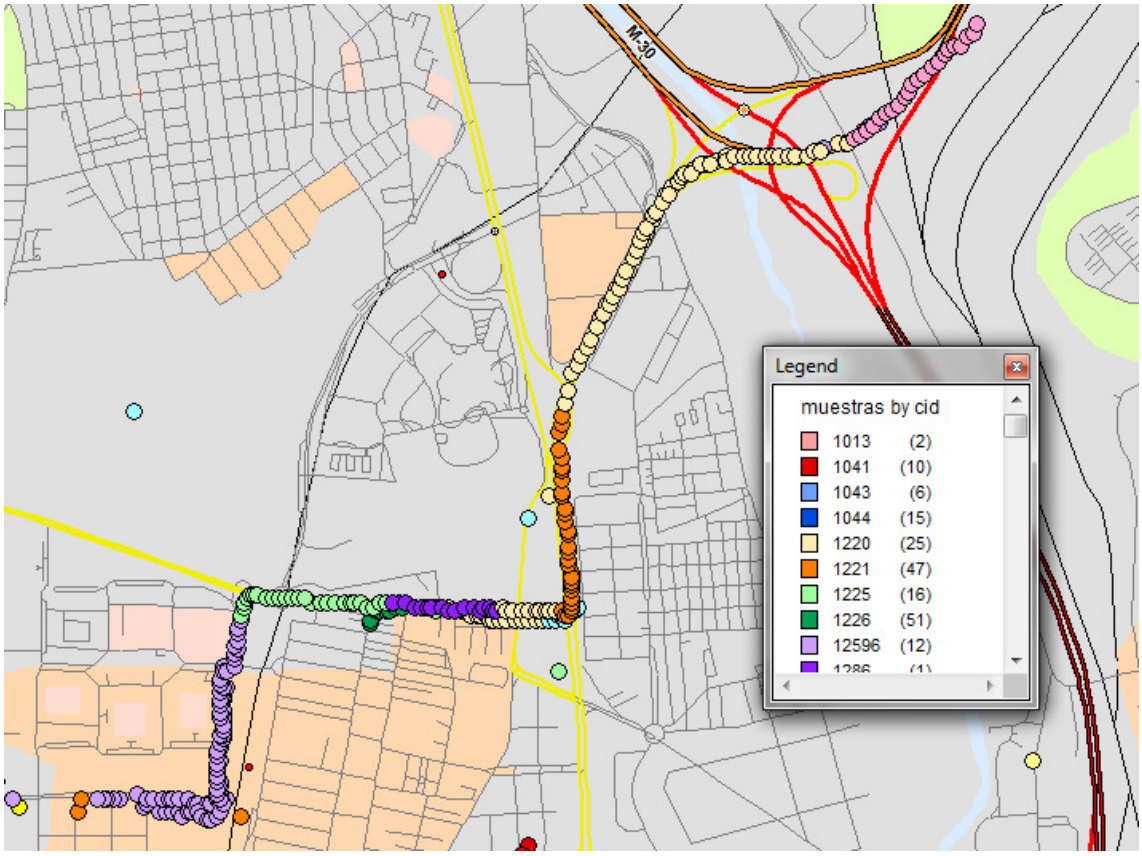

Figura 51: Estudio de muestras por CID (Cell Id)

Se representan con colores diferentes las muestras en función de a qué celda estén conectados para poder estudiar la secuencia de *Handovers* por la que atraviesa el usuario durante la captura de medidas. Es también información útil para el optimizador dado que de un simple vistazo puede observar si el recorrido marca o no un traspaso de llamada natural entre estaciones.

En el agregado de muestras puede observarse la distribución de celdas en las que más se conecta el usuario (figura 52), presentado por el identificador decimal CID. En el diagrama se puede concluir que la mitad de las conexiones se efectúa únicamente en 10 celdas.

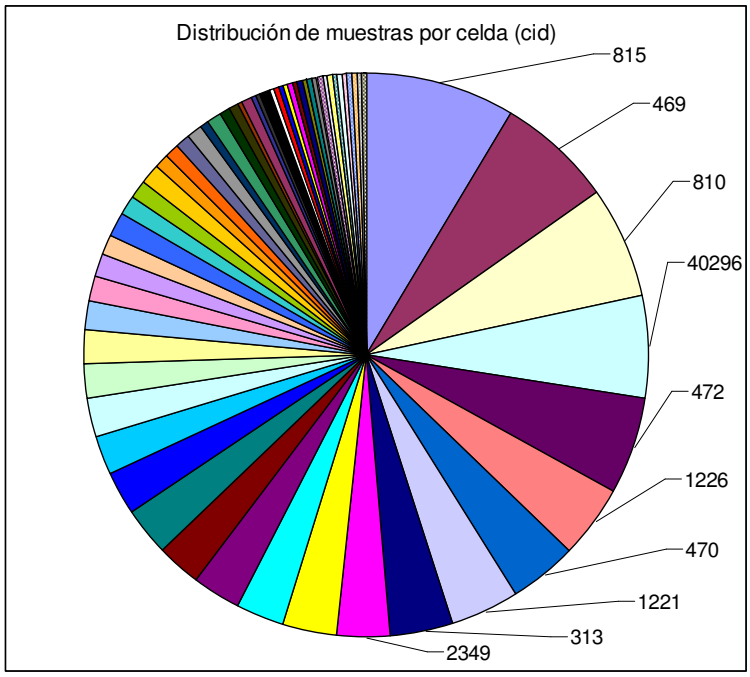

Figura 52: Diagrama de tarta de muestras por CID

5.2.4. Detección de zona de medida

En las dos primeras pruebas tratamos de verificar el comportamiento del *Proximity Alert*, indicando al móvil el mismo punto don distinto radio realizando dos pasadas en coche sobre la zona:

- **Prueba 1:** *ProximityAlert* con radio 300 metros.

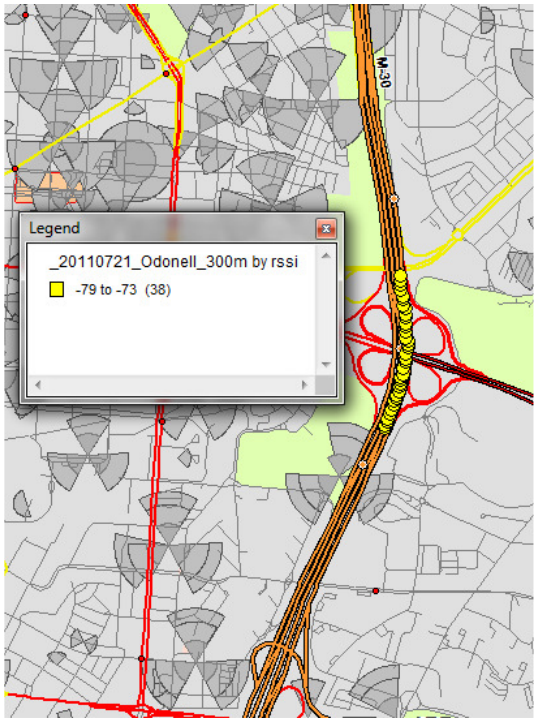

Figura 53: Centro Calle Odonell, radio 300m

- **Prueba 2**: *ProximityAlert* con radio 1200 metros.

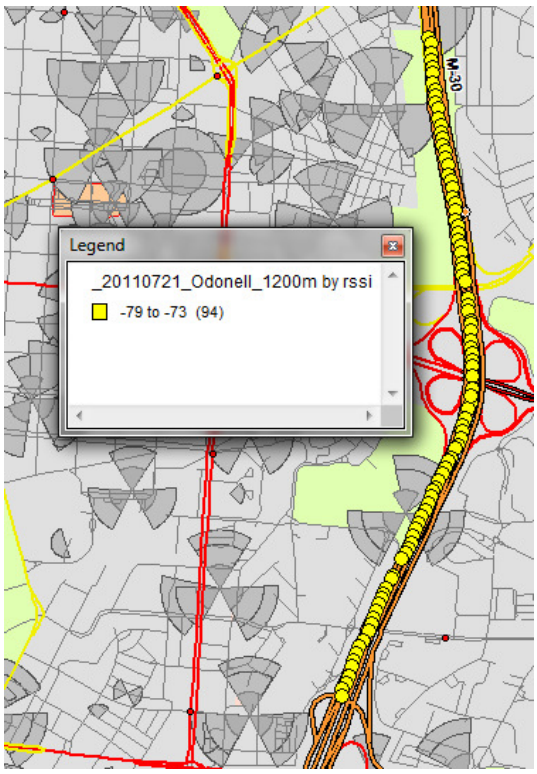

Figura 54: Centro Calle Odonell, radio 1200m

En estas dos pruebas en torno al puente de la calle Odonell de Madrid, comprobamos que la alerta de proximidad funciona de manera precisa con el GPS activado ya que 300 metros antes de llegar al centro de la región de medida comienza el terminal a capturara muestras. Justo 300 metros después, las medidas se cancelan. En la prueba 2 se ha realizado exactamente los mismos pasos pero indicando vía SMS al terminal que el radio es de 1200 metros. Queda probado así uno de los objetivos que nos marcamos desde el principio: la medida de cobertura bajo demanda configurable de forma remota.

- **Prueba 3:** *ProximityAlert* con radio 9000m: Desde C/Odonell hasta Tres Cantos

Es en esta prueba donde conseguimos cierta relevancia estadística indicando al terminal un área de medida de 9km. Ahora tenemos suficientes muestras de señal para detectar posibles zonas de escasos niveles de señal. En todo el tramo de la carretera M-30 (como cabía esperar pues se trata del centro de Madrid) la cobertura es buena. Sin embargo, aparecen tramos azules, en la carretera de Colmenar, con menor despliegue y con posibles zonas de cobertura débil. Estos resultados se muestran en la figura 55:

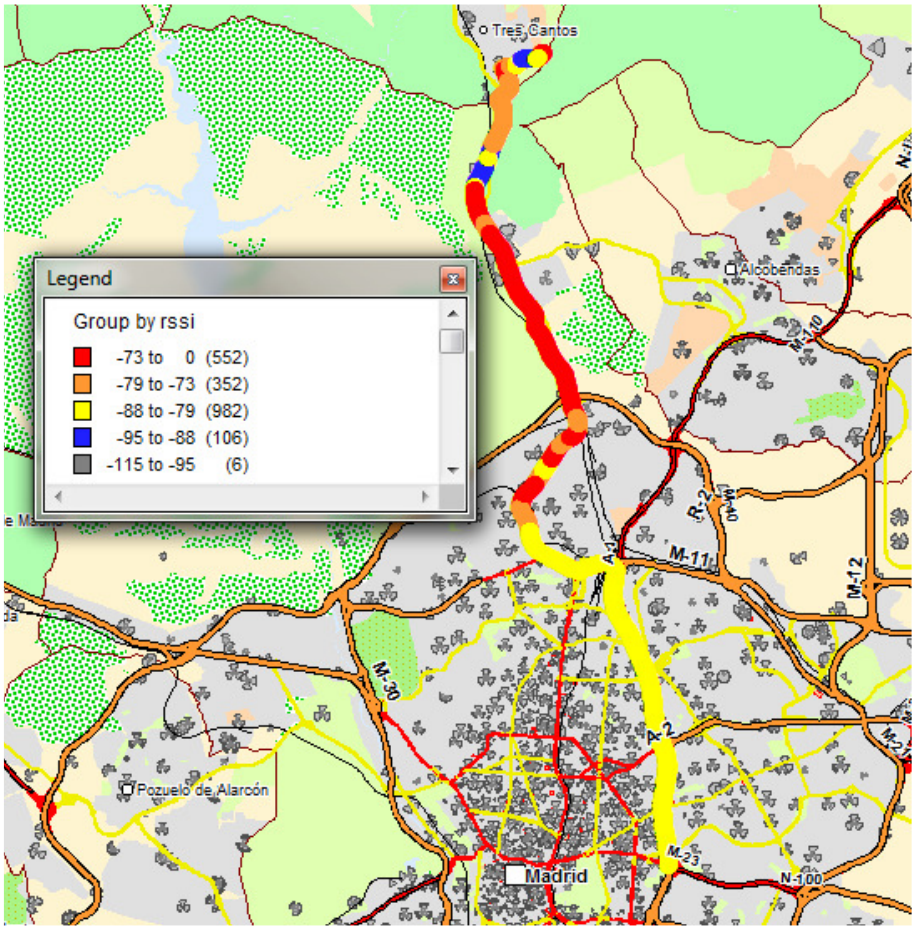

Figura 55: Centro Nudo Norte, radio 9000m

- **Prueba 4:** *ProximityAlert* con radio 40 metros:

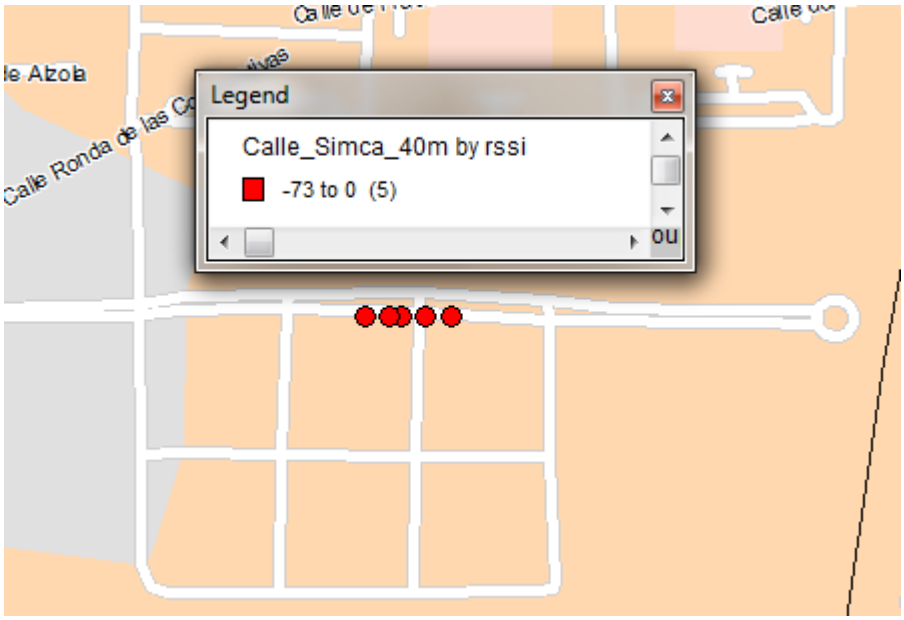

Figura 56: Centro C/ Simca, radio 40m

Para probar la precisión del sistema no sólo se eligen áreas grandes. En esta prueba se ha tomado como radio 40 metros y la aplicación no falla al tener que

arrancar y apagar las medidas en poco espacio de tiempo. El número de muestras es pequeño dado que como ya vimos, se ha establecido la resolución del GPS en 10 metros.

- **Prueba 5:** *ProximityAlert* con radio 80 metros y varias pasadas:

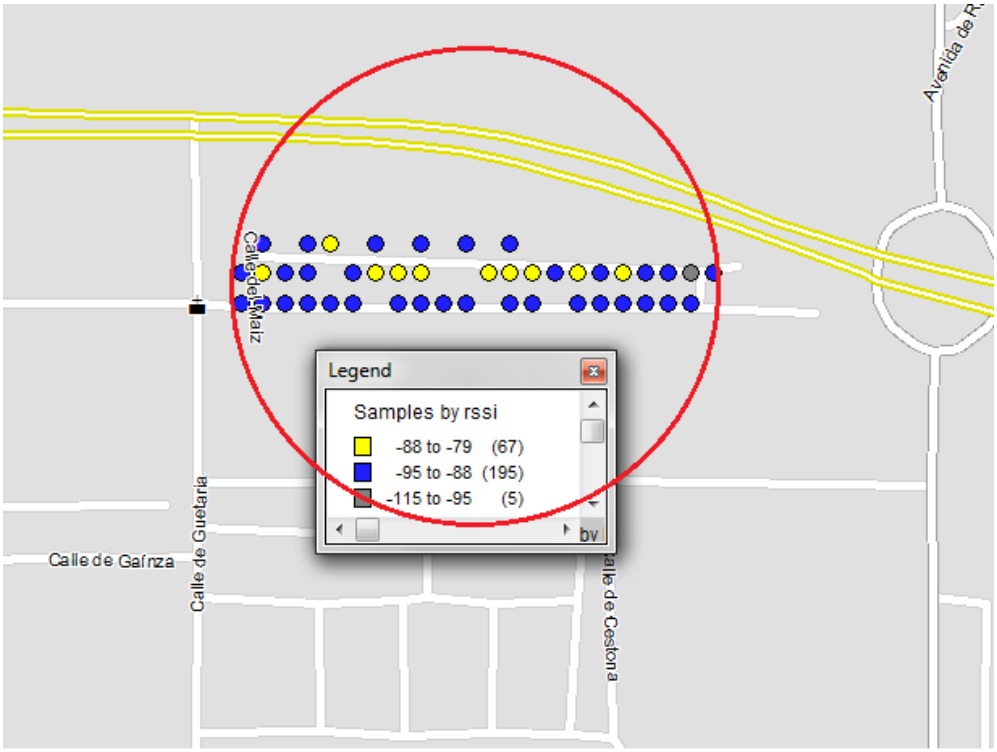

Figura 57: Radio 80 metros. Varias pasadas

El objetivo de esta prueba es comprobar la activación y desactivación de las medidas al entrar y salir del área de estudio. En este escenario se realizan 6 pasadas en la región indicada confirmando la captura selectiva de muestras.

5.2.5. Captura de llamadas caídas

Desde que el sistema estaba preparado para capturar llamadas caídas, en el terminal de pruebas se ha inicializado la monitorización en 25 días. En ellos se han detectado 100 llamadas caídas, que se presentan a continuación en la figura 58.

Esta es una de las partes clave del Proyecto ya que de estas muestras pueden sacarse muchas conclusiones. En la gráfica se presentan las caídas por nivel de señal en el momento de producirse. Aparecen muchas con buenos niveles de señal (rojos) dado que al principio el umbral de detección de llamada caída era de -60 dBm (casi cualquier llamada terminada se consideraba caída).

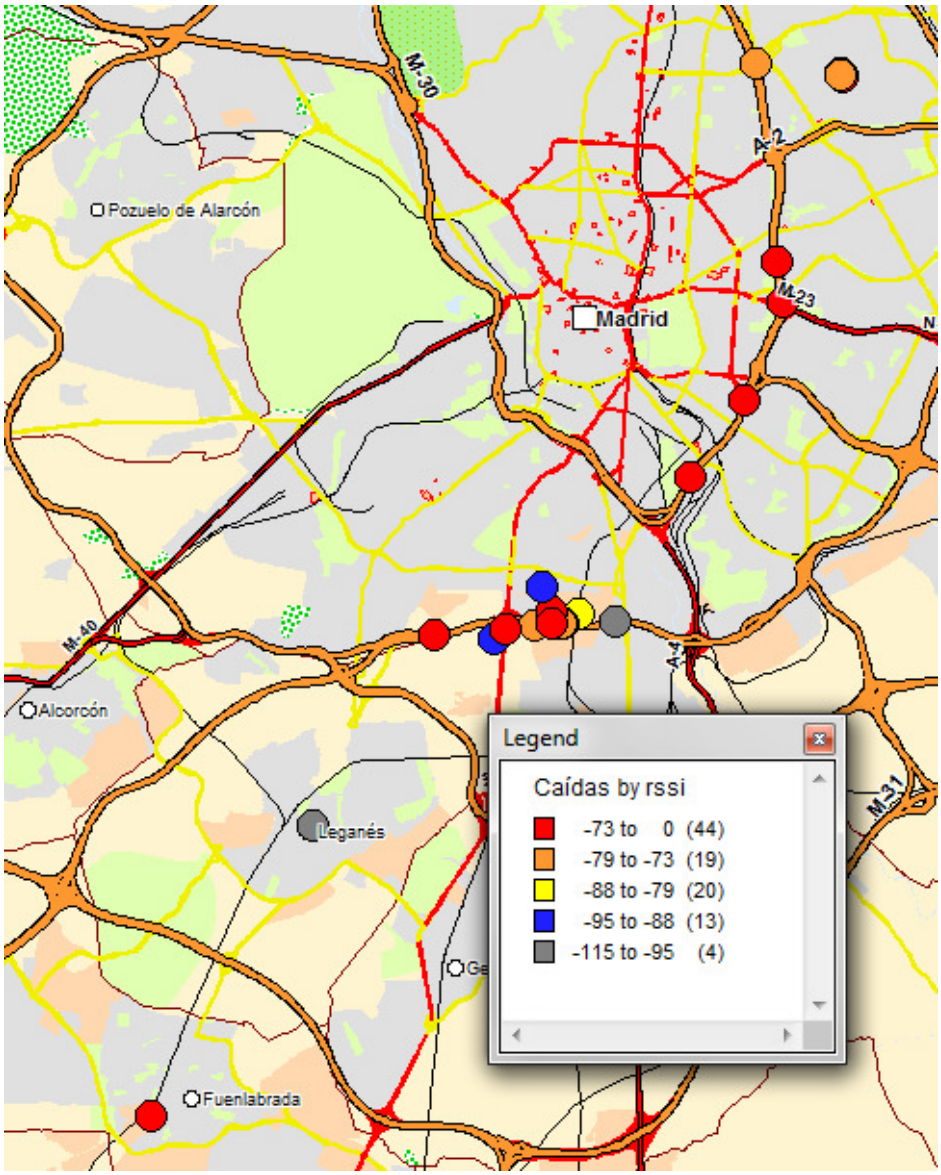

Figura 58: Llamadas caídas detectadas

Posteriormente el umbral de llamada caída se establece en -95 dBm, que ya es un valor considerable para poder producir caída en cualquier entorno, sobretodo Indoor. Con el mapa global de caídas, puede observarse el patrón de utilización de la red por parte del usuario. En este caso se observa actividad predominante en el distrito de Usera en Madrid y gran cantidad de muestras en distrito de Hortaleza. Además pueden verse algunas llamadas caídas en el traslado entre ambas zonas, concretamente en la Autopista M-30.

Con las muestras capturadas se observa sobretodo concentración de llamadas caídas con bajos niveles de señal en la zona de Usera, dado que la mayoría de las caídas se producen en interiores y además es una región con escasa cobertura (en estos casos la localización se basa en red, dado que en interiores no es posible utilizar GPS).

Simplemente con estas medidas el operador sería capaz de identificar las reclamaciones de un cliente que viviera en la zona y conocer casi en tiempo real qué nivel de cobertura se está ofreciendo.

5.2.6. Detección de zona de riesgo

En una primera fase, para la detección de zona de riesgo, se han realizado pruebas **bajando el umbral de señal** para la detección de llamadas caídas de tal modo que cualquier desconexión doble en menos de 45 segundos dispare el SMS de alarma. En las pruebas se realizaban llamadas seguidas al buzón de voz, para comprobar que tras la segunda desconexión se envíaba el SMS de lerta con el formato ya visto en 4.10:

Fecha;Hora;latitud;longitud;señal;LAC;CID;tipored

Y en pantalla se muestra la siguiente notificación:

Figura 59: Notificación de envío de alarma de zona de riesgo

No es necesario que el usuario sepa que se encuentra en zona de riesgo. De hecho esta notificación no debería existir, únicamente se presenta en la aplicación a modo informativo y para comprobar el correcto comportamiento de la aplicación en estos casos.

Una vez confirmada la generación de alerta, su notificación y envío de SMS correspondiente, en las pruebas siguientes se han detectado dos zonas de riesgo con el umbral de nivel de **señal en -95 dBm**, que es el valor final establecido en el sistema. Son las muestras presentadas con flechas en la siguiente imagen (figura 60)

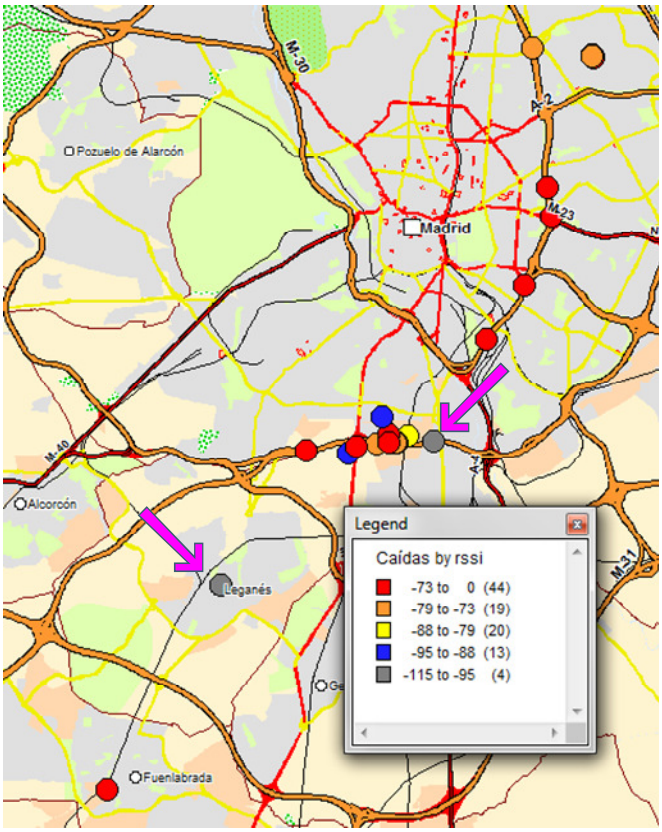

Figura 60: Detección de zona de riesgo

En la figura se representan dos puntos que identifican sendas zonas de riesgo tal y como hemos diseñado la alarma. En formato texto, la detección de ambas situaciones e la siguiente:

Fecha	Hora	lat	lon	rssi	lac	cid	type	mode
20110311	17:50:45	40,3297	-3,7645	-97	21021	3531	UMTS	IDLE
20110311	17:51:12	40,3297	-3,7645	-97	21021	3531	UMTS	IDLE
20110414	22:58:23	40,3647	-3,6964	-97	1004	1041	GSM	IDLE
20110414	22:58:59	40,3647	-3,6964	-97	1004	1041	GSM	IDLE

Podemos comprobar como se cumplen los requisitos de diseño para la detección de zona de riesgo, dado que ambas se producen por debajo de -95 dBm y observando la hora de los eventos, se confirma que en ambos casos la llamada se reintenta y vuelve a fallar en menos de 45 segundos. Inmediatamente después se construye el SMS de alarma que se envía al servidor como aviso de posible zona de riesgo.

6. Conclusiones y líneas de Trabajo Futuras

Tras la exposición de resultados de todas las pruebas realizadas llega la hora de hacer balance del trabajo realizado y de presentar las posibilidades de trabajo futuras basadas en el sistema implementado.

6.1. Conclusiones

En el **sistema de monitorización de calidad de señal**, hemos conseguido registrar en puntos geográficos concretos cierta información relevante que posibilita la realización de estudios más o menos complejos de cobertura y de eventos de llamada caída.

Éste era el principal **objetivo** que marcamos al comienzo del diseño. Hemos logrado "anclar" cada una de las muestras recogidas a **posiciones geográficas** que aportan un avance sustancial a los métodos de optimización clásicos que se basan en datos agregados de red con resolución de celda. Del conjunto de terminales que participan en el sistema distribuido de recolección de datos, es posible obtener un **modelo representativo** del comportamiento de una red de telefonía móvil extrapolando los resultados conseguidos desde el subconjunto de pruebas.

Desde el comienzo del Proyecto el objetivo de utilización del sistema era el del **Operador de Red**. Los resultados obtenidos de este tipo de estudios suponen una fuente de información muy potente para los procesos de optimización internos y de mejora constante de la calidad de servicio ofrecida. Con la instalación de esta aplicación (que no supera en peso los 100 KB) en un conjunto de terminales lo suficientemente amplio, el propietario de la red dispone de un "enjambre" de sondas distribuidas de forma aleatoria que capturan información de forma transparente y con bajo coste computacional, para aportar de forma agregada, el **estado actualizado de la calidad de la red**, y que además puede ser representado cartográficamente.

Pero no sólo se ha desarrollado la aplicación para funcionamiento en modo "free running", sino que el comportamiento de los terminales en la recolección de datos es regido totalmente por la **entidad central** de forma **remota,** cumpliendo así con otro de los objetivos planteados. En las pruebas realizadas se han llevado a cabo varios escenarios de configuración de medidas, indicando específicamente al terminal **qué zona concreta medir** desde el servidor central. La gobernabilidad del sistema por parte del operador evita captura de muestras por defecto o exceso dado que las muestras deben ajustarse a los intereses del propietario de la red. De forma remota pueden ordenarse medidas en zonas de especial interés por tratarse de regiones de nuevo despliegue, ferrocarriles, carreteras, detección de posibles fallos de red o escasa cobertura.

La otra funcionalidad clave del sistema es la detección de **llamadas caídas** y su registro, anotando para cada una de ellas posición y contexto en que se producen. Como vimos en la fase de pruebas, el sistema es capaz de conocer con exactitud en qué puntos geográficos suceden estos eventos y de forma agregada pueden detectarse zonas con alta densidad de puntos que impliquen posibles fallos de red. Además, no sólo se almacenan las llamadas caídas sino todas aquellas finalizaciones de llamada por debajo de un cierto umbral de señal, lo que aporta una fuente de información extra en el conjunto de muestras: ya que son llamadas con alto riesgo de fallo.

De la extracción de llamadas caídas pueden realizarse gran variedad de informes como son la diferenciación del comportamiento de la red en GSM y UMTS, acumulación de caídas en torno a una celda (lo que puede indicar posibles errores de la BTS, ya sean SW o HW). Además, en un conjunto de muestras, ya sea por medida de cobertura manual o automática, puede identificarse la ruta de celdas por las que transcurre una comunicación, dado que uno de los parámetros que se capturan en cada muestra es el identificador de celda (LAC-CID). Con esta información puede seguirse la secuencia de *handovers* de una llamada y desde el punto de vista del optimizador detectar posibles incoherencias en dicha secuencia.

La otra funcionalidad que permite completar el cumplimiento de objetivos marcados en las especificaciones, es la **detección de zona de riesgo**. Cada terminal que monitoriza las llamadas caídas es capaz de reaccionar ante situaciones excepcionales de comportamiento de la red. Aunque el motivo sea un fallo del terminal, cada vez que se producen 2 llamadas caídas en menos de 45 segundos, de forma **inmediata** se comunica mediante un SMS al servidor central la información en que se produce el evento. Esto es muy importante porque es posible que una estación haya caído (por ejemplo debido a fallos de energía) y ésta sea la manera más rápida de detectarlo, antes incluso de que lo hagan los grupos de Operaciones y Mantenimiento de red (grupos O&M). La detección inmediata de incidencias (y su posterior resolución) repercute directamente en la **calidad percibida** por parte del usuario.

Todas estas funcionalidades se han desarrollado en la plataforma **Android**. En la elaboración de la aplicación cliente que forma parte del sistema, se ha construido desde cero y tras la fase de aprendizaje del lenguaje, una aplicación completa que no sólo responde a las especificaciones planteadas sino que presenta al usuario una interfaz gráfica con la que interactuar y poder realizar, entre otras cosas, medidas de cobertura bajo demanda. Las pruebas se han realizado con dos terminales de fabricantes distintos probando así el correcto funcionamiento en más de un dispositivo, ambos con versiones 1.6 de Android.

En resumen, El Proyecto se ha basado en la implementación de un sistema capaz de capturar en distintos terminales de una red, información de utilidad para el operador de red, de forma transparente a los usuarios y geolocalizada. Esta filosofía no sólo se limita a la detección de los eventos vistos en el Proyecto sino que las posibilidades son muy amplias, como veremos a continuación en la sección de trabajos futuros.

6.2. Estudios futuros

A partir de la filosofía de geolocalización de información de usuario y ampliando el ámbito de acción de la aplicación desarrollada, el siguiente paso es crear una "*suite*" completa de captura y análisis de datos que aportan directamente los usuarios de una red móvil. Sin duda el primer paso sería desarrollar el equivalente en el resto de plataformas o al menos en las más extendidas (Apple iOS, RIM, Windows Phone).

Algunos de los estudios que pueden realizarse a partir de estos conceptos son los siguientes y que de forma natural serían el siguiente paso en la mejora del sistema propuesto:

1. **Comparativa de terminales:**

   A partir del IMEI, dado que este identificador incorpora el modelo del terminal y el fabricante, se pueden extraer conclusiones de comportamientos radio de cada uno de ellos por separado. En este par de ejemplos presentamos una distribución de comportamiento en recepción de señal y llamadas interrumpidas por fabricante:

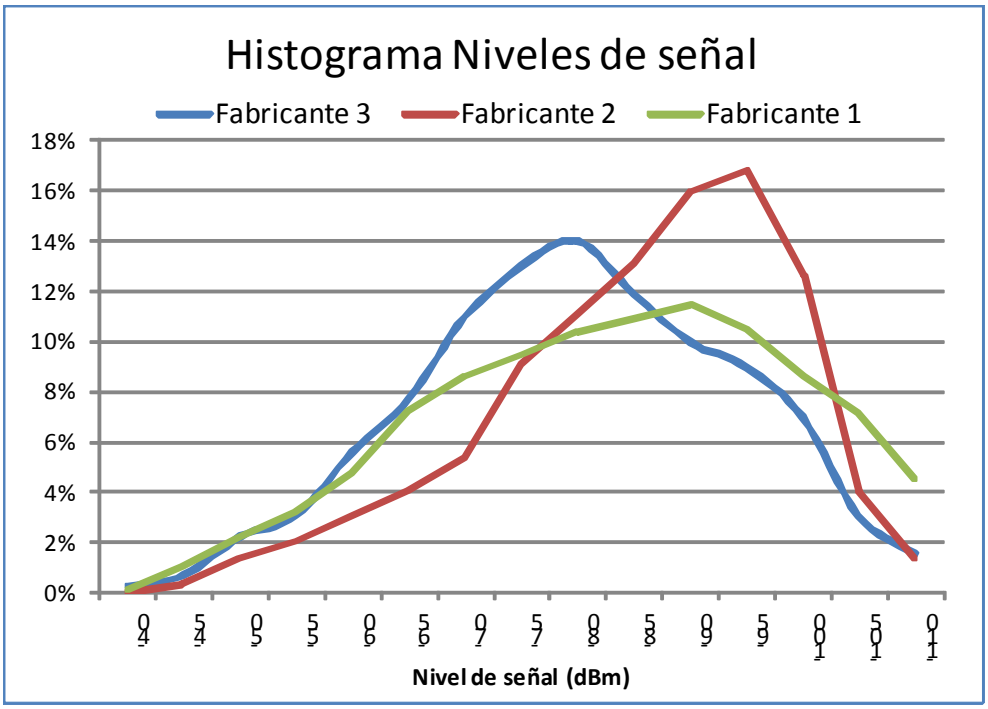

Figura 61: Histograma recepción señal por fabricante

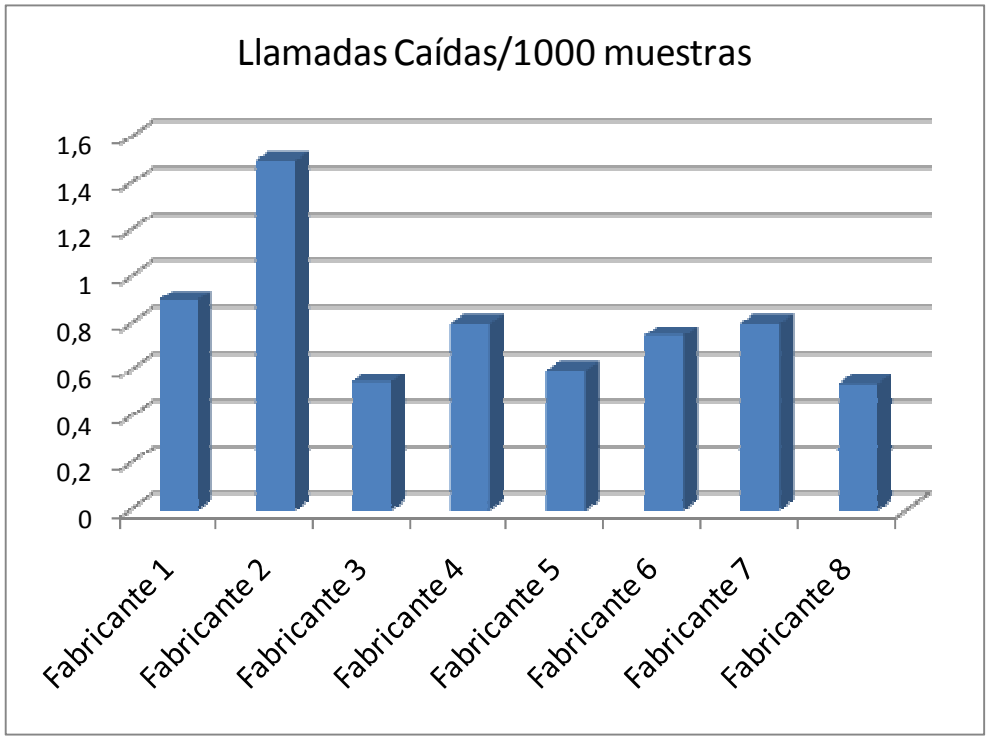

Figura 62: Número de caídas relativo por fabricante

En ambas gráficas puede apreciarse que, en promedio, un terminal del fabricante 3 presenta mejores características radio que los demás[4]. Las aplicaciones de este tipo de estudios son muy diversas en función del interés de los resultados. Puede servir a un fabricante concreto para la mejora de un modelo de terminal con fallos de fabricación para su corrección. Además es una fuente de información interesante para el operador que puede comprender (si su red empeora) si es debido al incremento del parque de un determinado modelo con un mal *performance* de calidad.

2. Estudio de otros indicadores:

En el sistema hemos decidido focalizar el estudio en llamadas interrumpidas por considerar éste, el evento más perjudicial desde el punto de vista de la percepción de usuario. Sin embargo el uso cotidiano del servicio de telefonía móvil comprende gran cantidad de indicadores que afectan a la calidad y que sin duda son un buen punto de partida para trabajos futuros. Con ciertas mejoras del sistema pueden obtenerse capturas de llamadas bloqueadas/fallidas, conexiones de datos (para estudios de volúmenes y tipos de servicio), Handovers (dónde se producen y dónde fallan), etc.

3. Estudio de indicadores en movilidad:

En el Proyecto hemos considerado posición pero no velocidad. Ésta es fácil de obtener y permite conocer el contexto en que se realiza la llamada en

[4] Las gráficas por fabricante no hacen referencia a situaciones reales. Son datos ficticios que sirven para representar de forma gráfica el potencial de la recolección de datos basada en IMEI.

cuanto a movilidad se refiere. El estudio de optimización de red asociado a eventos de llamada caída es muy diferente si ésta se produce en estático o en movimiento. En este segundo caso suelen intervenir varias estaciones y el problema puede localizarse en el traspaso de llamada (*Handover*). Son dos escenarios muy diferentes pero con la captura de la velocidad pueden realizarse informes de Caídas Vs. Velocidad.

4. Estudio de indicadores por franja horaria:

Uno de los parámetros que se capturan siempre es el tiempo. De esta forma, al recoger todas las muestras de los dispositivos que formen parte del "enjambre" distribuido del sistema, pueden realizarse informes de comportamiento de red por franja horaria. Las llamadas caídas no se distribuyen igual a lo largo del día, existen patrones de movilidad en horarios laborables y de ocio que precisan de estudios concretos para identificar en que zonas geográficas se dan los eventos de fallo en la llamada (por ejemplo en el metro en hora punta).

5. Seguimiento por usuario:

Otro de los estudios que permitiría el sistema es el seguimiento de usuarios concretos ya sean particulares o de empresa. Siempre cumpliendo todas las leyes de protección de datos y de privacidad, o con el consentimiento de los clientes, un operador puede tracear a determinados usuarios que denuncian fallos en el servicio y acotar así los puntos geográficos concretos donde se producen o en qué estaciones está conectado cuando detecta degradación en el servicio.

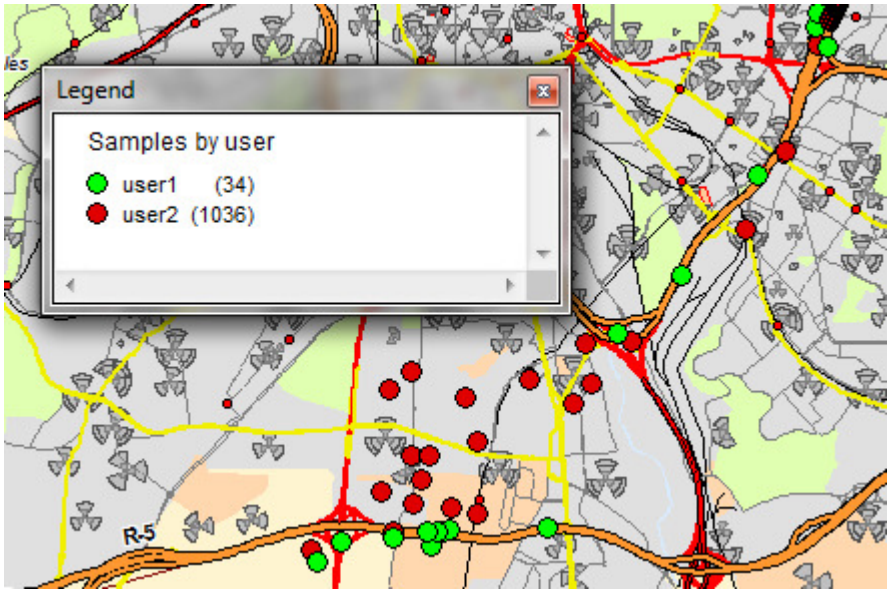

Figura 63: Seguimiento de muestras por usuario

6. Reporte interactivo de incidencias:

Relacionado con el punto anterior (reclamaciones de clientes), uno de los puntos en que podría mejorarse el sistema es la inclusión de una sencilla interfaz gráfica para el reporte interactivo de incidencias. Como vimos en el

apartado 2.4, los usuarios pueden comunicar al propietario de red determinados problemas de servicio normalmente por teléfono o por correo electrónico. Esto hace difícil el procesado automático de la información para su posterior análisis por lo que una buena solución sería la de incorporar una sencilla mejora en la aplicación.

La Interfaz presentaría un mapa donde el usuario seleccionaría un punto geográfico concreto donde detecta fallos de servicio. A continuación se abrirían una serie de menús desplegables para que el usuario introduzca la información necesaria para la reclamación: tipo de fallo (Cobertura, llamada caída, llamada no realizable, conexión de datos lenta, etc.) y la tecnología si la conoce (GSM/UMTS). Esta información se encapsula en pocos caracteres que pueden enviarse al operador incluso por SMS, sin coste alguno para el cliente (aunque se propondría un número máximo de quejas al mes para evitar uso abusivo de la red en este aspecto).

Al final, el servidor central posee gran cantidad de información respecto a reclamaciones de cliente que puede ser procesada fácilmente y sin intermediarios como son los operadores de "*call centers"*. El inconveniente es el que vimos ya en la sección 2.4, y es que se trata de información totalmente subjetiva.

Anexo 1: Presupuesto

1.- Autor:
Daniel Delgado Vico (Ingeniería de Telecomunicación)
2.- Departamento:
Ingeniería Telemática
3.- Descripción del Proyecto:

- Titulo	**Monitorización de la calidad de señal en redes móviles basada en Android**	
- Duración (meses)	**7**	
Tasa de costes Indirectos:	**5.073,55**	**20%**

4.- Presupuesto total del Proyecto (valores en Euros):

30.441,32

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)
					0,00
Delgado Vico, Daniel		Ingeniero Senior	3	4.289,54	12.868,62
Delgado Vico, Daniel		Ingeniero	4	2.694,39	10.777,56
					0,00
					0,00
		Hombres mes	**7**	**Total**	23.646,18

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Teléfono Sony Ericsson X10	360,00	100	7	60	42,00
Teléfono Orange Boston	275,00	100	7	60	32,08
Portátil ASUS K52F	900,00	100	7	60	105,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	179,08

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Licencia Mapinfo® Professional		1.450,00
Kilometraje		47,50
Facturación Telefónica		45,00
	Total	1.542,50

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	23.646
Amortización	179
Subcontratación de tareas	0
Costes de funcionamiento	1.543
Costes Indirectos	5.074
Total	30.441

Tabla 1: Formato estándar de presupuesto Universidad Carlos III de Madrid [15]

Anexo 2: Referencias

- [1] Holma, Harri. *WCDMA for UMTS: radio access for third generation Mobile communications.* Ed: John Wiley & Sons .2001
- [2] Hernando Rábanos, José María. *Comunicaciones móviles GSM*. Ed: Fundación Airtel. 1999
- [3] Ableson, Frank – Collins, Charlie – Sen, Robi. *Android: Guía para desarrolladores*. Ed: Anaya. 2009
- [4] L. Murphy, Mark. *Begining Android.* Ed: Apress. 2009
- [5] Rogers, Rick – Lombardo, John – Mednieks, Zigurd & Meike, Blake. *Android Application Development*. Ed: O'Reilly. 2009
-
- [6] DiMarzio, J.F. *Android: A programmer's guide*. Ed: McGraw Hill. 2009
- [7] Android Developers. http://developer.android.com/index.html
- [8] http://www.gartner.com/technology/about.jsp
- [9] http://www.openhandsetalliance.com
- [10] http://www.eclipse.org/
- [11] http://developer.android.com/guide/topics/fundamentals/activities.html
- [12] https://sites.google.com/site/a1ayuda/
- [13] http://www.pbinsight.com/welcome/mapinfo/
- [14] http://es.scribd.com/doc/55978747/33/PAGING
- [15] Formato Presupuesto Universidad Carlos III de Madrid:
- http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_1.xlsx

Anexo 3: Términos

- **AGPS**: *(Assisted GPS).* Mejora de respuesta de GPS basada en un servidor terrestre de "asistencia".
- **API**: *(Application Programming Interface).* Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizados por otro software como una capa de abstracción.
- **ASCII**: *(*American Standard Code for Information Interchange*).* Codificación binaria estándar de carácteres.
- **BSC**: *(Base station controller).* Elemento de red encargado de trasportar señalización y tráfico entre las BTS's y la red móvil.
- **BTS**: *(Base Transceiver Station).* Estación base que sirve como punto de acceso a la red móvil.
- **CID***: (Cel Id).* Identificador de Celda (BTS).
- **CN**: *(Core Network).* Es el núcleo de la red.
- **CMT**: *(Comisión del Mercado de Telecomunicaciones).* Entidad de derecho público cuya misión es salvaguardar las condiciones de competencia en el mercado de las telecomunicaciones
- **CRM**: *(Customer Relationship Management).* Modelo de gestión basada en la orientación al cliente
- **DCR**: *(Drop Call Rate).* Relación entre llamadas Caídas y llamadas iniciadas. Mejor cuanto menor sea este indicador.
- **DDMS**: *(Dalvik Debug Monitor Server).* Herramienta del emulador de Android.
- **FTP**: *(File Transfer Protocol).* Protocolo de red para la transferencia de archivos basado en la arquitectura cliente-servidor.
- **GIS***: (Geographic Information System).* Sistema de manipulación de datos geográficos.
- **GPRS**: *(General Packet Radio Service).* Extensión del sistema GSM para la transmisión de datos por paquetes. También conocida como 2.5G
- **GPS**: *(Global Positioning System).* Sistema global de localización por satélite basado en la sincronización de señales enviadas por estos.
- **GSM** (2G): *(Global System for Mobile Communications).* Norma de transmisión y recepción para la telefonía celular que inicialmente se canalizó en 900MHz.
- **Handover**: *Traspaso de llamada de una BTS a otra sin interrupción de la misma.*
- **IMEI**: *(International Mobile Equipment Identity)*. Identificador de fabricante y modelo de terminal móvil.
- **KPI**: *(Key Performance Indicator).* Son indicadores clave que presentan métricas definidas para cuantificar objetivos que reflejan el rendimiento de una actividad.
- **LAC***: (Local Área Code).* Identificador de área. Comprende un conjunto de nodos B /BTS.
- **LTE***:(Long Term Evolution).* Evolución del sistema UMTS ó 4G.
- **RAN***:(Radio Access Network).* Elementos de red comprendidos hasta la capa jerárquica superior (Core).
- **RNC**: *(Radio Network Controller).* Elemento de red de la arquitectura WCDMA encargado del control de las estaciones base 3G o Nodos B.
- **SAC**: *(Servicio de Atención al Cliente)*
- **SGSN**: *(Serving GPRS Support Node).* Elemento de red encargado de la gestión de paquetes de datos en redes de telefonía.

- **SMS**: *(Short Message System).* Sistema que permite el envío de mensajes cortos de hasta 140 caracteres entre dispositivos móviles.
- **SQL**: *(Structured Query* Language*).* Es un lenguaje declarativo de acceso a bases de datos relacionales.
- **TCH**: *(Traffic Channel).* Canal de tráfico contenido en las tramas de comunicación del sistema GSM.
- **TRX**: *(Transmitter/Receiver).* Elemento hardware de la BTS encargado de la transmisión y recepción de las señales de radiofrecuencia.
- **Toast**: *Modo de notificación de Android mediante ventana emergente.*
- **UE**: *(User Equipment).* Siglas que identifican a cada uno de los usuarios (dispositivos) finales de una red.
- **UMTS** (3G): *(Universal Mobile Telecommunication System).* Evolución del sistema de telefonía GSM que incluye como principal mejora la banda ancha de datos móvil.
- **XML**: *(eXtensible Markup Language).* Metalenguaje universal basado en etiquetas.